





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

Alejandra Mancilla <sup>1</sup>, Mario García-Valdez <sup>1,\*</sup>, Oscar Castillo <sup>1</sup> and JJ Merelo-Guervós <sup>2</sup>

<sup>1</sup> Tijuana Institute of Technology; {alejandra.mancilla, mario, ocastillo}@tectijuana.edu.mx

<sup>2</sup> 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.

**Copyright:** © 2022 by the authors. Submitted to *Symmetry* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 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, now 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:  $\text{near}(2) = .92$  and  $\text{near}(50) = .40$ . 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 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 route

[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 globally, avoiding to be trapped 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. These algorithms can be broadly grouped into evolutionary algorithms (EAs) [22] and swarm intelligence (SI) [23], as well as other categories; popular EAs are Genetic Algorithms (GAs) [24,25], Genetic Programming (GP) [22], and Differential Evolution (DE) [26], while examples of (SI) [23] are particle swarm optimization (PSO), [27], Grey Wolf Optimization (GWO) [28] and Aquila Optimizer (AO) [29]. Moreover, examples of other categories are the Harmony Search (HS) [30] algorithm, that is inspired by how jazz musicians improvise when playing with others, and finally, the arithmetic optimization algorithm (AOA) [31] 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 [32,33]. Wagner and Hagnas [34] 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 [35] for evolving the parameters of all the MFs of a coupled-tank liquid-level control system. Astudillo et al. [36] 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 [37], 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 [38,39], 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

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

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. In our previous work [38,39], 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 track 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 route so that this research could solve similar problems.

The main contribution of this work is to propose, through the use of population-based metaheuristics, 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 [40,41]. The front-wheel can steer in the axis normal to the plane of motion, the steering angle is  $\delta$  (see Figure 1), The position of the midpoint of the rear-wheel is given by the coordinates  $x_r$  and  $y_r$ . The heading  $\theta$  is the angle of the link between the two wheels and the  $x$  axis. We follow the model describe in [42], with the differential constraint:

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

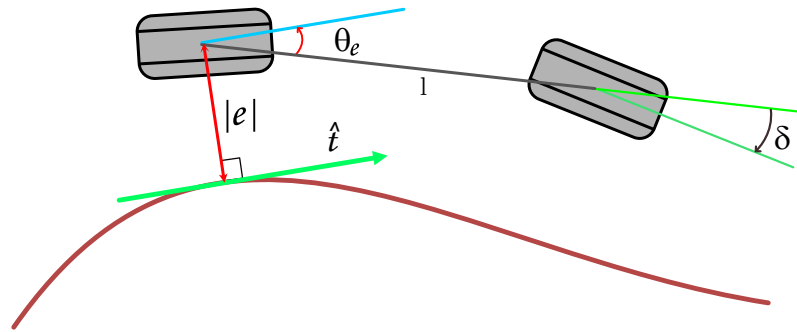
The controller selects the steering angle  $\delta$  with a value between the limits of the vehicle  $\delta \in [\delta_{min}, \delta_{max}]$  and a desired velocity  $v_r$  again limited by  $v \in [v_{min}, v_{max}]$ . The heading rate  $\omega$  is related to the steering angle by

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

and we can simplify the heading dynamics to

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

138 .



**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.  $\theta_e$  is the difference between the tangent at the nearest point in the path and the the heading  $\theta$ . The output of the controller is the heading rate  $\omega$ , from which we calculate the steering angle  $\delta$  of the front wheel.

Now we explain the path tracking method described by Paden et al. [42]. 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 [43], and the feedback is a function of the nearest point on the reference path given by

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

and the tracking error vector is

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

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

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

The error  $e$  is the cross product of the two vectors

$$e = d_x \hat{t}_y - d_y \hat{t}_x \quad (8)$$

The heading error uses the angle  $\theta_e$  between the robot's heading vector and  $\hat{t}$

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

## 148 2.2. Fuzzy Controllers

149 To design a fuzzy controller from the model we just described, we must consider  
 150 the magnitude of the scalar value  $e$ , the distance from the rear wheel to the route's  
 151 closest point. This error is positive if it is on the right and negative on the left of the  
 152 route. Another input variable is the angle  $\theta_e$  defined between the heading vector and the  
 153 tangent vector of the route. This angle can also be negative or positive depending on  
 154 the vehicle's position concerning the route; the controller output is the heading rate  $\omega$ ,  
 155 which allows us to calculate the steering angle  $\delta$ . The target velocity of the robot will be  
 156 constant, so we will not control the velocity  $v$  using the fuzzy controller we will use the  
 157 following simple proportional controller instead

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

158 In this work, we will compare two fuzzy models, one with three MFs, which was  
 159 proposed in our previous work [39], and a new model using five MFs. We describe both  
 160 fuzzy systems next.

### 161 2.2.1. Fuzzy Model with Three Membership Functions

162 We now model the same input and output variables mentioned above as fuzzy  
 163 variables using three membership functions. Each variable has the same fuzzy values:  
 164 high negative (hi\_neg), low (low), and high positive (hi\_pos). Depending on the  
 165 case, the sign indicates whether it is to the right or left of the path. We use trapezoidal  
 166 functions for the "high" fuzzy term (positive and negative) and a triangular function for  
 167 the "low" value.

168 We consider a total membership in extreme values (depending on each variable's  
 169 domain). Because of this, we ignore the trapezoidal parameters' values outside the value  
 170 domain. In total, we could modify up to 33 parameters for all membership functions. We  
 171 propose keeping the extreme parameters outside the domain and the central point of the  
 172 triangular function fixed. The center of the triangular functions will always be at 0 since  
 173 it is the minimum error possible. We kept the fuzzy rules symmetrical, representing how  
 174 a human driver would adjust the steering wheel when driving. The rules are presented  
 175 in Table 1. We have nine rules, to establish a relation between all the possible inputs and  
 176 output values.

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

Rule 1:	If $\theta_e$ is	hi_neg	and $e$ is	low	then $\omega$ is	hi_pos
Rule 2:	If $\theta_e$ is	hi_pos	and $e$ is	low	then $\omega$ is	hi_neg
Rule 3:	If $\theta_e$ is	low	and $e$ is	low	then $\omega$ is	low
Rule 4:	If $\theta_e$ is	hi_neg	and $e$ is	hi_neg	then $\omega$ is	hi_pos
Rule 5:	If $\theta_e$ is	hi_pos	and $e$ is	hi_pos	then $\omega$ is	hi_neg
Rule 6:	If $\theta_e$ is	hi_pos	and $e$ is	hi_neg	then $\omega$ is	low
Rule 7:	If $\theta_e$ is	hi_neg	and $e$ is	hi_pos	then $\omega$ is	low
Rule 8:	If $\theta_e$ is	low	and $e$ is	hi_pos	then $\omega$ is	hi_neg
Rule 9:	If $\theta_e$ is	low	and $e$ is	hi_neg	then $\omega$ is	hi_pos

### 177 2.2.2. Fuzzy Model with Five Membership Functions

178 We now add two more membership functions to the previous fuzzy model, in-  
 179 tending to have a finer grain of adjustment. Adding more functions also adds more  
 180 complexity to the rules, and now we have even more parameters to tune. Instead of  
 181 having just two values for each type of error hi and low we add a middle value,  
 182 represented by the medium membership function. The knowledge in the fuzzy rule  
 183 base is now more complex than before, with 25 rules. In this case, defining the rules was  
 184 not done by simply specifying a driver's knowledge. We needed to adjust the rules by  
 185 running several simulations. The rules are presented in Table 2.

**Table 2.** Proposed fuzzy rules for the basic controller with three membership functions.

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

### 2.3. Tuning of Parameters Using a Genetic Algorithm

As we mentioned earlier, the proposed fuzzy controllers we proposed in the previous Section 2.2 need to be tuned to give acceptable results on a simulation. In this section, we describe our proposed method for MFs parameter optimization. First, we need to establish which parameters we are going to keep fixed and which parameters we are going to tune. The following two sections explain the parameter selection and the overall technique.

#### 2.3.1. Parameter Tuning for Three MFs configuration

In this configuration, we modified the input and output variables' parameters symmetrically, varying the minimum extremes of the trapezoidal functions

$$\mu_{trap}(x) = \max \left( \min \left( \frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right) \quad (11)$$

where the trapezoidal begins to descend/ascend and the point where it reaches zero.

We varied these parameters symmetrically  $(a, b, d, e, g, h)$ ; in the case of the triangular function (1), the aperture we symmetrically optimized  $(c, f, i)$  the parameters as shown in Table 3. To keep the symmetry, we just repeat the same parameter value on both sides of a triangular function or on the values of the trapezoidal. Some values are used only to make the function broader or narrower at the bottom. In this case, we parameterize the controller with just nine variables, and these are the parameters that the GA will optimize. That we will explain in Section 2.3.

#### 2.3.2. Parameter Tuning for Five MFs configuration

In a similar fashion as before, we need to define which parameters will be tuned for the five MFs controller. Again we keep the MFs symmetrical around zero. In this case,



**Table 3.** Nine parameter configuration for three MFs fuzzy controller.

Variable	Linguistic Value	MF	Parameters
$\theta_e$	high negative	$\mu_{trap}$	$[-100, -100, -a, -a + b]$
$\theta_e$	low	$\mu_{tria}$	$[-c, 0, c]$
$\theta_e$	high positive	$\mu_{trap}$	$[a + b, a, 100, 100]$
error	high negative	$\mu_{trap}$	$[-100, -100, -d, -d + e]$
error	low	$\mu_{tria}$	$[-f, 0, f]$
error	high positive	$\mu_{trap}$	$[d + e, d, 100, 100]$
omega	high negative	$\mu_{trap}$	$[-8, -8, -g, -g + h]$
omega	low	$\mu_{tria}$	$[-i, 0, i]$
omega	high positive	$\mu_{trap}$	$[g + h, g, 8, 8]$

we just needed ten variables to parameterize the controller. To achieve this, we kept the parameters of  $\omega$  fixed. The parameters are illustrated in Figure 4.

**Table 4.** Ten parameter configuration for five MFs fuzzy controller.

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

### 2.3.3. Parameter Tuning for Three MFs configuration

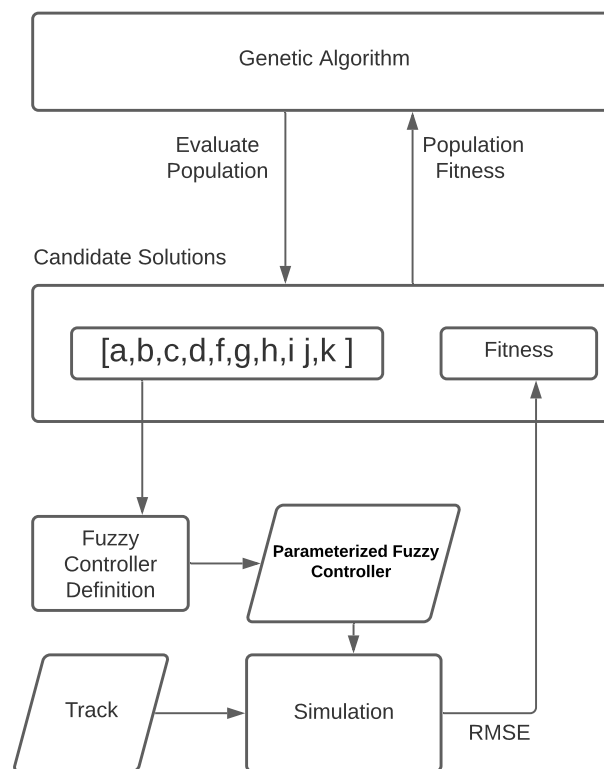
Another essential aspect to consider when tuning the above parameters is the range of values that each parameter can have. Normally, we keep all the parameters in the same range when using a population-based metaheuristic. In our previous work, we compared two ranges of values for the first controller, with 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. These values are shown in Table 5.

**Table 5.** 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	$[0, 1]$

#### 2.3.4. Genetic Algorithm Optimization

As a population-based metaheuristic, the GA 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 chromosome, here implemented as a list of objects of type `float`. To evaluate each solution, we need to generate an instance of the fuzzy controller. The membership functions are defined using the parameters included in the chromosome. Once created, the fuzzy controller is passed as a parameter, together with the tracks 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. The GA will not select the candidate solutions with the worst fitness for reproduction.



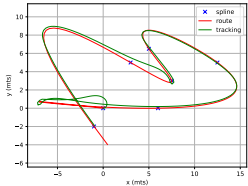
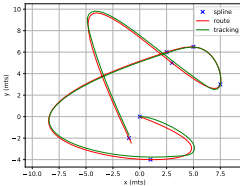
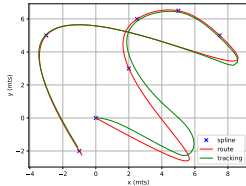
**Figure 2.** For each individual in the population, a controller is created with the parameters, this controller is then tested by running one or more simulations. The RMSE of the tracking is considered as the fitness for that particular candidate solution. This process is repeated for each member of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

In our previous work [39] we evolved the controllers with 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 track used for its evolution. We found that the controller with only three MFs could follow new unseen tracks, 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 [44]. Noticing this, we added a list of three tracks to evaluate the fitness for the experiments, which is the average RMSE of the three simulations. We ran experiments with one and three tracks. We defined each track using cubic splines over a list of coordinates; this is a common approach in the literature [45]. The tracks and the parameters to define them are shown in Table 6. The first track was used on previous work and was taken from the library of [46], this track 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 track “M”. The other tracks are called “A” and “S” and have smoother curves, with long straight segments and different curvatures. All tracks have seven anchor points for the cubic spline.

**Table 6.** Tracks used for fitness evaluation, they are defined by cubic splines with the parameters shown below each plot.

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

### 2.3.5. Experimental Setup

We implemented the algorithm in Python using the DEAP [47] library for the GA implementation. We used the scikit-fuzzy<sup>1</sup> library to code the fuzzy inference systems. Finally, we modified a fork of the PythonRobotics repository [46], adding the odeint library from the SciPy package<sup>2</sup> to integrate the system of ordinary equations.

All experiments ran with the same parameters for the GA. The only obvious difference was the size of the chromosomes because this is the same as the number of parameters we need to optimize. As we mentioned earlier, we proposed two controllers, one with three membership functions; we are going to call this controller 3MF, and the other controller 5MF because it has five membership functions. For details on these controllers, see Section 2.2. After several preliminary experiments, we settled for a configuration with a population size of 50 with 20 generations. We used single-point crossover with 0.3 probability. We opted for tournament selection with a tournament size of three. For mutation, and because we are using continuous values, we selected a Gaussian mutation with  $\mu = 0.0$  and  $\sigma = 0.2$ . Finally, the probability for each gene to be mutated was 0.2. Each individual’s fitness was the RMSE 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 track or did not finish near the final point of the track. 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 our results against the controller in [42], with the following control law defined as

$$\omega = \frac{v_r \mathcal{K}(s) \cos(\theta_e)}{1 - \mathcal{K}(s)e} - (k_\theta |v_r|) \theta_e - \left( k_e v_r \frac{\sin(\theta_e)}{\theta_e} \right) e, \quad (12)$$

the parameters of the simulations and the canonical controller we are comparing against are summarized in Table 7.

<sup>1</sup> <https://github.com/scikit-fuzzy/scikit-fuzzy>

<sup>2</sup> <https://github.com/scipy/scipy>

**Table 7.** Simulation and controller parameters.

Parameter	Value
Wheel-base	$l = 2.5$
Steering limit	$ \delta  \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>.

### 3. Results

This section shows the results of running the GA optimization of parameters for the two controllers described in the previous sections. We have the 3MF and 5MF controllers by running the GA optimization 30 times, using the average of the three tracks to establish the fitness. Moreover, we also ran a GA for both controllers using only the most challenging track (Track M) for the evaluation. The descriptive statistics of these results are shown in Table 8. As expected, and because the optimization is non-deterministic, there are a few outliers at both extremes of the performance. However, the deviation is higher when using only one track for the GAs evaluation. Nevertheless, the 5MF controller has the lower RMSE in both experiments. The problem we found and mentioned earlier is that the 5MF controller trained with a single track can not follow the path of the other tracks; this is not the case with the 3MF controller. For this reason, we will only consider the controller optimized with three tracks for further comparison. In the last column, we have the average execution time in seconds, for as expected, evaluating with three tracks is more time-consuming. However, we can also observe that it takes more time to evolve the 5MF controller.

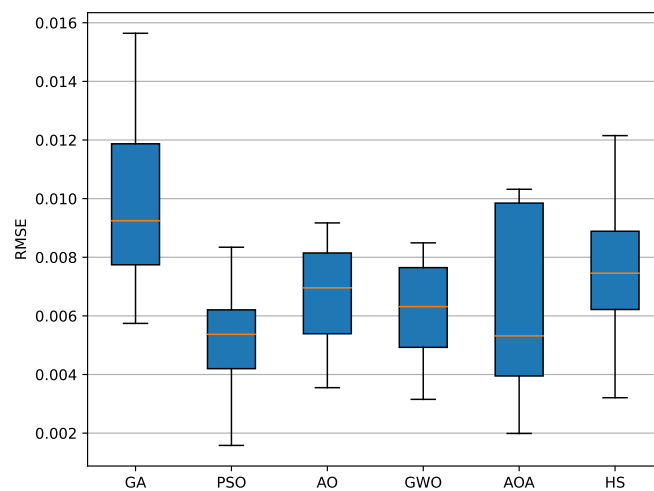
**Table 8.** Descriptive statistics for 30 executions of the GA with an evaluation of one and three tracks.

Evaluation	Controller	Average RMSE	Standard Deviation	Median	Average Exec. Time (s)
Three tracks	MF3	0.4321	0.071	0.4136	1283s
Three tracks	MF5	<b>0.0156</b>	0.031	0.0091	2664s
Single track	MF3	0.6831	0.1096	0.6866	533s
Single track	MF5	<b>0.0330</b>	0.1030	0.0055	640s

We selected the best controllers found in the 30 experiments and compared them against a controller using the control law in Equation 12. This results are presented in Table 9. Again the controller 5FM has the lower RMSE on all individual tracks. The 3MF controller has a similar RMSE when we compare it against the control law.

**Table 9.** RMSE of the best controllers in all three tracks.

Track	Control law	3MF	5MF
Route M	2.003	0.820	<b>0.003</b>
Route A	0.014	0.016	<b>0.005</b>
Route S	0.521	0.162	<b>0.008</b>

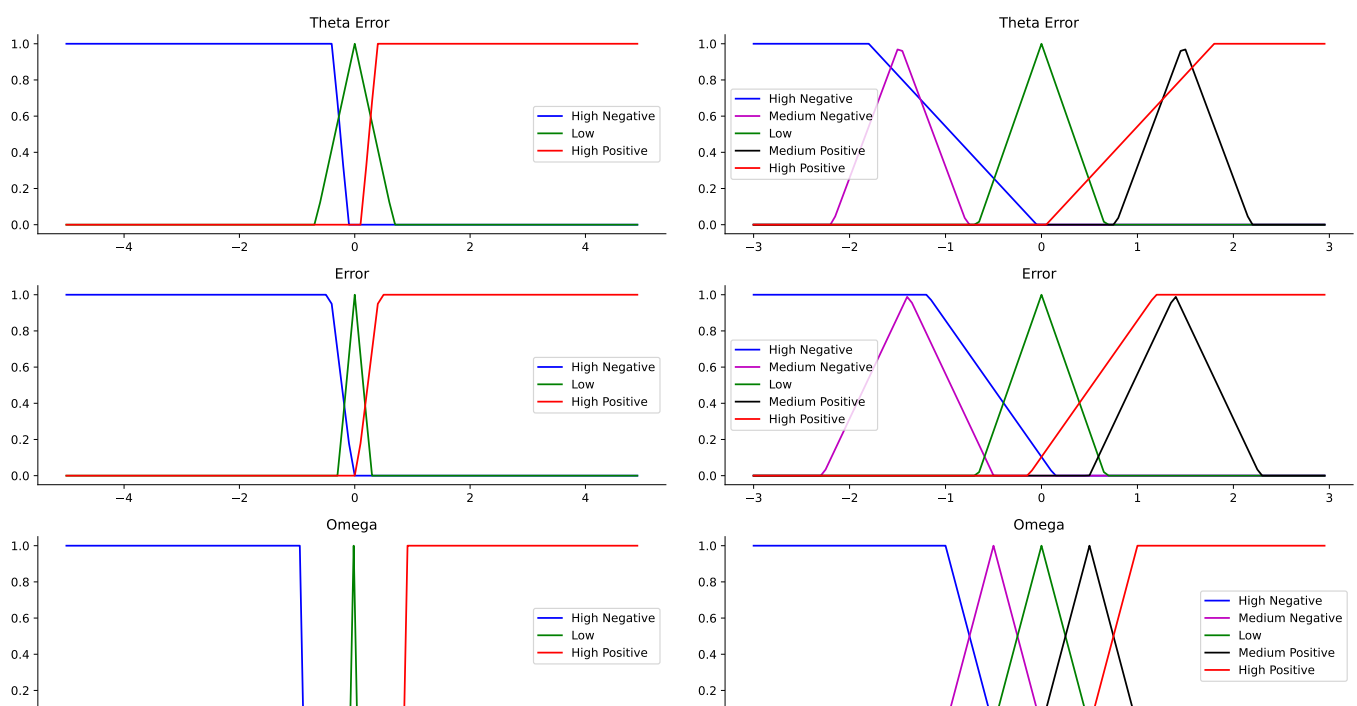


**Figure 3.** For each individual in the population, a controller is created with the parameters, this controller is then tested by running one or more simulations. The RMSE of the tracking is considered as the fitness for that particular candidate solution. This process is repeated for each member of the population.

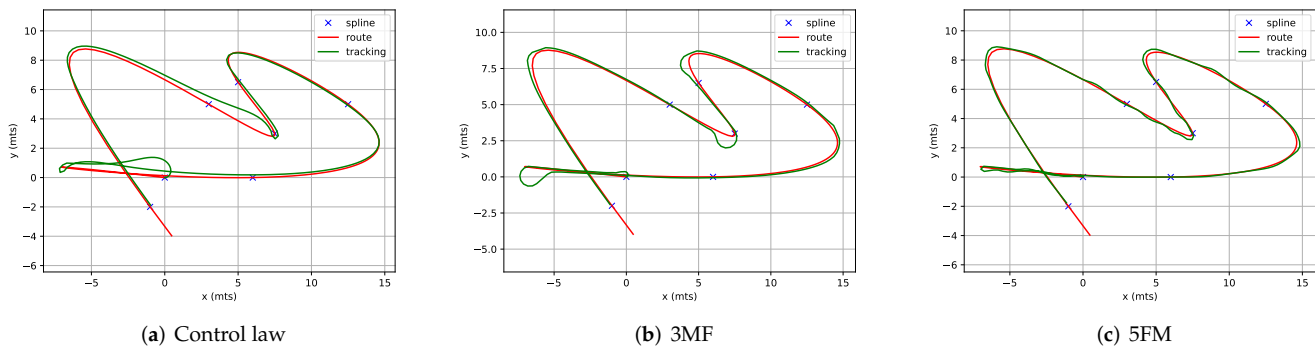
**Table 10.** Wilcoxon rank-sum test between algorithms, showing p-values for  $H_1 : A < B$

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

When we observe the optimized MFs for both controllers, 3MF in Figure 4(a) and 5MF in Figure 4(b), we can see that they are similar on the low and high MFs, for both the  $\theta_r$  and  $e$ , moreover, in each of them both variables are very similar. With only omega having very different parameters. We can also see that the values of  $\omega$  remain fixed on 5MF, because we kept those parameters fixed.

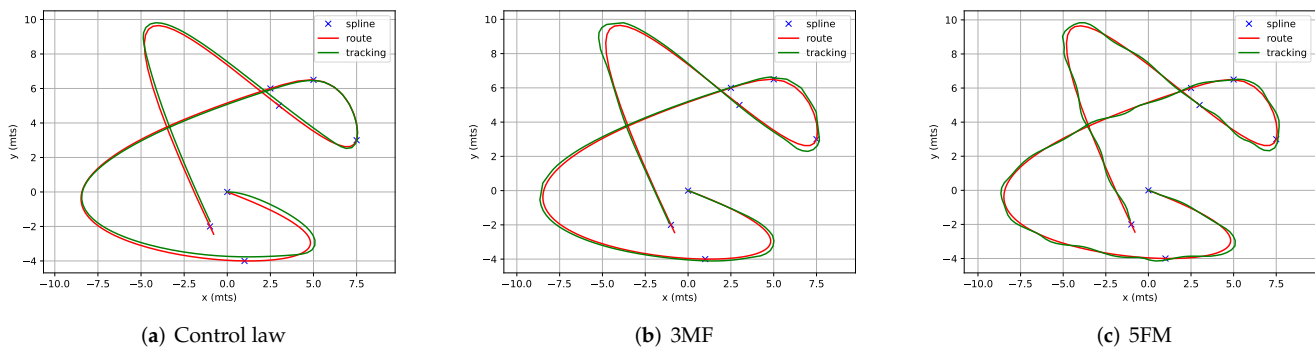


302 We now show a plot of the tracks and the robot's movement following the track.  
 303 First, we have the track "M" (see Figure 5), in which all controllers have problems at  
 304 the beginning with a curve that is very sharp to the left and then a sharp U-turn to go  
 305 back to the start. We can see that the 3MF controller follows the path with less zig-zag  
 306 than controller 5MF, which keeps the tracking closer to the path but with noticeable  
 307 zig-zagging.



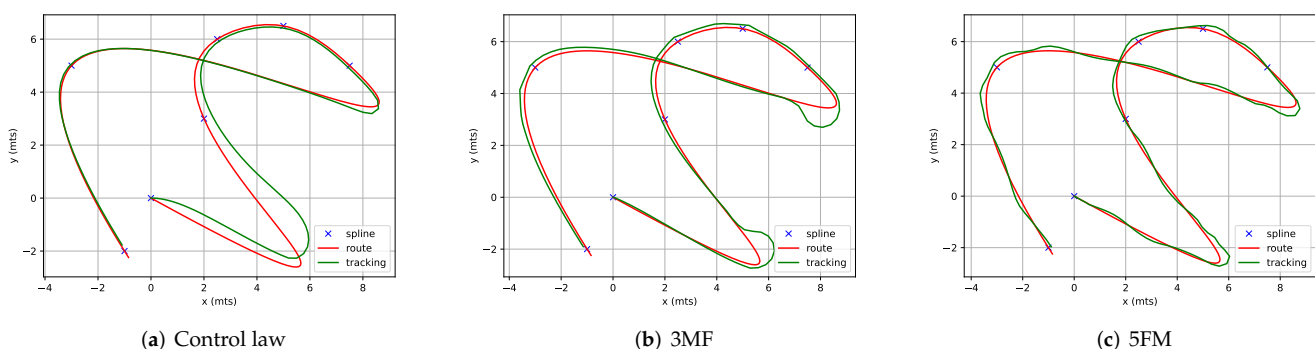
**Figure 5.** Plot for the best simulations on Track M.

308 The plots in Figure 6 show the "A" track. This time all controllers closely follow the  
 309 path. Again controller 5MF has a noticeable zig-zagging but manages to have the lower  
 310 RMSE of the three.



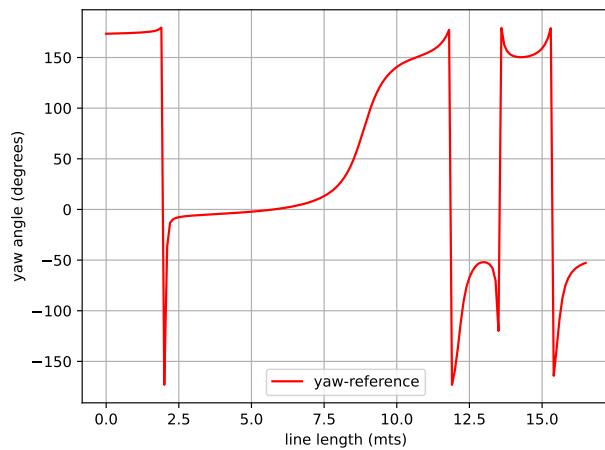
**Figure 6.** Plot for the best simulations on Track A.

311 The results of Figure 7 highlight the overall behavior of the three controllers. The  
 312 control law takes more time to reach the path when it deviates from the reference because  
 313 it has smoother steering, but once it is over the track,  $e$  does not increase. That is by  
 314 design because one of the conditions is that a small initial tracking error will remain  
 315 small. On the other hand controller 3MF, does not deviate much from the reference, has  
 316 a smoother control than the 5MF, but it has a higher RMSE.

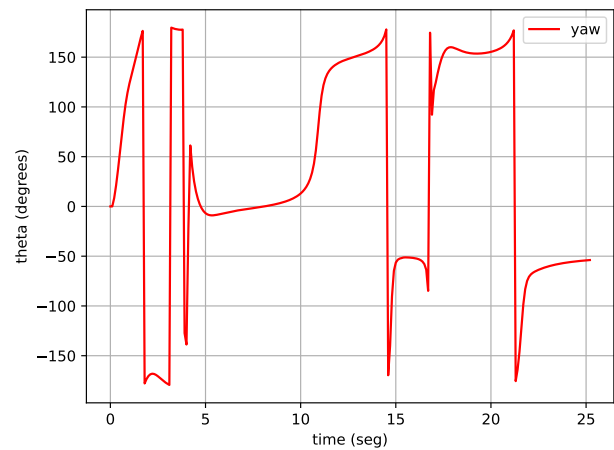


**Figure 7.** Plot for the best simulations on Track S

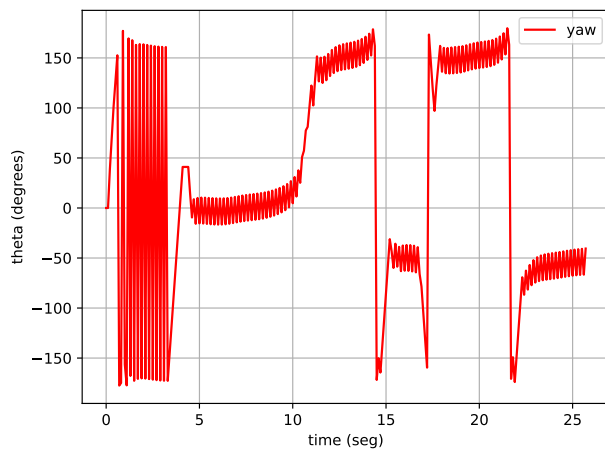
317 Finally, in Figure 8 we can have a better look at the steering motion of each of the  
 318 controllers concerning the yaw of the “M” path. Here we can see the advantage of the  
 319 control law. In this case, the 3MF controller has a very high oscillation, which is not  
 320 desirable in this type of controller. The same is observed in the 5MF controller but to a  
 321 lesser degree; this could be related to how these controllers evolved, giving only weight  
 322 to  $e$ .



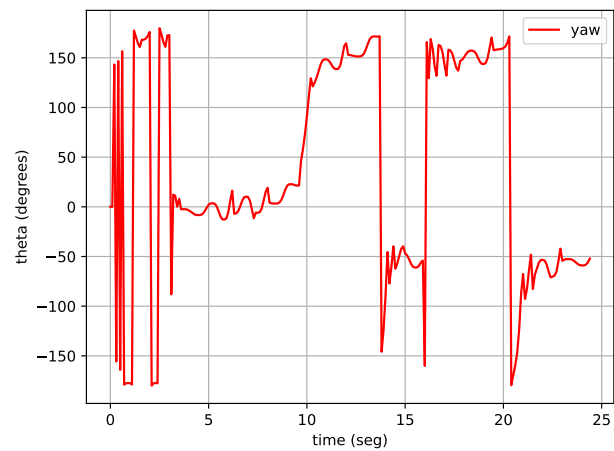
(a) Yaw along the reference path  $s(\gamma)$  in meters.



(b) Yaw of the control law as it was tracking the reference path, in seconds.



(c) Yaw of the 3MF controller.



(d) Yaw of the 5MF controller.

**Figure 8.** Comparison of the yaw (steering) for each of the controllers as it was tracking the reference path “M”, in seconds.

#### 323 4. Discussion

324 In this paper, we have proposed a method using a GA metaheuristic for evolving  
 325 fuzzy controllers by optimizing its membership function’s parameters. The fuzzy con-  
 326 troller, aimed at autonomous path tracking, using rear-wheel feedback, was optimized  
 327 by running simulations and measuring the RMSE of the distance between the robot’s  
 328 position and a reference path. We compared two controllers with distinct fuzzy inference  
 329 systems, using three MFs for each variable and a second with five MFs. Adding more  
 330 MFs and, consequently, more fuzzy rules added more complexity to the knowledge  
 331 base, adding more granularity to the representation. We have shown that this change  
 332 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  $\omega$ . 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.

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

- 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.
- 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.
- Driankov, D.; Hellendoorn, H.; Reinfrank, M. *An introduction to fuzzy control*; Springer Science & Business Media, 2013.
- 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.
- King, P.J.; Mamdani, E.H. The application of fuzzy control systems to industrial processes. *Automatica* **1977**, 13, 235–242.
- Passino, K.M.; Yurkovich, S.; Reinfrank, M. *Fuzzy control*; Vol. 42, Citeseer, 1998.
- Yang, X.; Moallem, M.; Patel, R.V. An improved fuzzy logic based navigation system for mobile robots. Proceedings 2003 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453). IEEE, 2003, Vol. 2, pp. 1709–1714.
- Driankov, D.; Saffiotti, A. *Fuzzy logic techniques for autonomous vehicle navigation*; Vol. 61, Physica, 2013.
- 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.
- 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.
- Bi, M. Control of Robot Arm Motion Using Trapezoid Fuzzy Two-Degree-of-Freedom PID Algorithm. *Symmetry* **2020**, 12, 665. doi:10.3390/sym12040665.
- 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.; Vitor, 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. *RLAI-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. Back, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*; Oxford university press, 1996.
23. Kennedy, J. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*; Springer, 2006; pp. 187–219.
24. Holland, J.H.; others. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*; MIT press, 1992.
25. Eiben, A.E.; Smith, J.E. Genetic algorithms. In *Introduction to evolutionary computing*; Springer, 2003; pp. 37–69.
26. 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.
27. Clerc, M. *Particle swarm optimization*; Vol. 93, John Wiley & Sons, 2010.
28. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Advances in engineering software* **2014**, *69*, 46–61.
29. 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.
30. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: harmony search. *simulation* **2001**, *76*, 60–68.
31. 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.
32. 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.
33. 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.
34. Wagner, C.; Hagra, 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.
35. 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.
36. 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.
37. 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.
38. 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.
39. 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.
40. 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.
41. 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.

42. 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.
43. Samson, C. Path following and time-varying feedback stabilization of a wheeled mobile robot. *International Conference on Control, Automation, Robotics and Vision*, 1992, 1992.
44. Tan, P.N.; Steinbach, M.; Kumar, V. *Introduction to data mining*; Pearson Education India, 2016.
45. 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.
46. 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.
47. Fortin, F.A.; De Rainville, F.M.; Gardner, M.A.G.; Parizeau, M.; Gagné, C. DEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research* **2012**, *13*, 2171–2175. Publisher: JMLR. org.