Neural Computing and Applications
https://doi.org/10.1007/s00521-018-3514-1

S.I. : ADVANCES IN BIO-INSPIRED INTELLIGENT SYSTEMS

Control structure for a car-like robot using artificial neural networks and genetic algorithms

Camilo Andrés Cáceres Flórez[1] · João Maurício Rosário[1] · Dario Amaya[2]

Received: 16 December 2017 / Accepted: 27 April 2018

Abstract
The idea of improving human's life quality by making life more comfortable and easy is nowadays possible using current technologies and techniques to solve complex daily problems. The presented idea in this work proposes a control strategy for autonomous robotic systems, specifically car-like robots. The main objective of this work is the development of a reactive navigation controller by means of obstacles avoidance and position control to reach a desired position in an unknown environment. This research goal was achieved by the integration of potential fields and neuroevolution controllers. The neuro-evolutionary controller was designed using the (NEAT) algorithm "Neuroevolution of Augmented Topologies" and trained using a designed training environment. The methodology used allowed the vehicle to reach a certain level of autonomy, obtaining a stable controller that includes kinematic and dynamic considerations. The obtained results showed significant improvements compared to the comparison work.

Keywords Neuroevolution · Artificial neural network · Genetic algorithm · Control strategy · Non-holonomic wheeled robot · Car-like robot

1 Introduction

Nowadays, due to the evolution of transport systems and intelligent systems, the requirement of autonomous transportation systems has increased, creating an important area of study for mobile robots and several control schemes that works under real conditions. Likewise, mobile robotic development and the improvement of new automation technologies and data processing have created the need to develop control systems that improve the autonomy of mobile robots in different work environments.

In the mobile robotics field, trajectory, position, and orientation controls have been the key development topics, applying techniques such as PID controllers, neuronal networks, and fuzzy logic controllers, specifically in coupled constrained models, like the car-like robot system. The navigation controller development for wheeled devices with kinematic constraints, like the non-holonomic robots, is significantly more complex than controllers designed for unconstrained robots.

In the following work, a constrained and non-holonomic mobile car-like robot is studied and controlled. Also, its navigation through an unknown environment filled with obstacles makes the controller task even more challenging, reason why the implementation of many obstacle avoidance techniques are required. In this study case, the dynamics of the system are included in a simplified way to make even more realistic the controller training.

In addition, some artificial intelligence techniques allow several kinds of algorithms to achieve example-based learning, Also a fitness function like neuroevolution methodology allows the neural network population

✉ Camilo Andrés Cáceres Flórez
camilocf@fem.unicamp.br

João Maurício Rosário
rosario@fem.unicamp.br

Dario Amaya
dario.amaya@unimilitar.edu.co

[1] Mechanical Engineering School, Integrated Systems Department, Universidade Estadual de Campinas – UNICAMP, Rua Mendeleiev, 200, Campinas, São Paulo CEP 13083-860, Brazil

[2] Faculty of Engineering, Mechatronic Engineering Program, Universidad Militar Nueva Granada – UMNG, Carrera 11 #101-80, Bogotá, Cundinamarca, Colombia

Published online: 11 May 2018

evolution using the algorithm as a method to optimize the network parameters [1]. One method of neuroevolution used commonly is the algorithm known as Neuroevolution of Augmenting Topologies (NEAT), which improves the neural network's topology, achieving thus better performance in some control tasks, in comparison to some traditional fixed topology neural networks [1–3].

There are many NEAT techniques and implementations, used in areas like robotics and control of mechanical systems, like the work of [4], where the NEAT algorithm was used to find a constrained optimal controller for the legs of a quadruped robot. A similar case is shown by Wen et al. [5], where a muscular-skeletal arm neurocontroller was developed using the NEAT algorithm. Another supplementary documented case is the video games field, where a neural network was used as a decision-making system for a strategy game [6]. Also, some other learning methods are applied in the mechatronics field, like control of different kinds of systems with artificial neural networks [7] or using fuzzy logic controllers [8–10].

With respect to autonomous robots, there are cases where different mobile robot types are controlled by means of fuzzy logic [11, 12], artificial neural networks [13], neuro-fuzzy algorithms [14] or neuroevolutive algorithms [15].

Traditional methods for mobile robots reactive navigation are a set of rules like the Bug 0, Bug 1 and Bug 2 algorithms, presented in [16]. Those algorithms do not heed the robot kinematic constraints, making some decisions very hard to follow by adding additional maneuvers to the robot motion; this makes them non-efficient algorithms for constrained robots.

Recent developments prove the good results of the popular idea of using soft computing techniques for autonomous robot navigation; in [17] the results of a brief survey of current techniques for path planning in realistic conditions were mostly soft computing techniques and statistical methods. An example of soft computing techniques usage for a mobile robot full control is shown in [18] by spiking neural networks and Hebbian learning implementation, demonstrating a control strategy with obstacle avoidance for holonomic robots inside unknown environments. Two similar works [19, 20] propose hybrid artificial intelligence usage, the first case a Neuro-Fuzzy algorithm and second one an Evolutionary Fuzzy control for mobile robots; in both cases, the navigation is through unknown environments, avoiding obstacles by reading the sensors information mounted on the robot chassis.

According to brief state of art presented above, the usage of soft computing algorithms and the need of including the robot physical behavior in the controller design algorithms, an unknown environment reactive navigation method based on soft computing techniques is proposed; it includes the robot kinematic constraints, dynamics, and control theory. This work is the consecutive part of [21]; however, in this case, the obstacle avoidance method has been significantly improved and the robot dynamics are also added to the model, increasing this proposal profoundness and strengthening its structure.

Therefore, the objective of this paper is to propose a control strategy for a non-holonomic car-like robot, capable of avoiding obstacles and reaching a target point inside unknown environment by means of neural networks, genetic algorithms, and potential fields. The objective is achieved by including the system behavior in the embedded algorithm and testing the chosen non-holonomic car-like robot in some random environments.

This paper is organized as follows: In Sect. 2, the car-like robot kinematics and dynamics are presented, including the PID motor controllers. In Sect. 3, the neuroevolutive controller design together with the position controller and the obstacle avoidance method are developed. In Sect. 4, the algorithm behavior is presented with the complete controller integration and its results. Finally, the conclusions of the study are presented.

2 Kinematics and dynamics

The kinematic and dynamic models of the chosen car-like robot are fundamental for the development of this research. Those models allow the controller training and evolution for the studied mobile robot in a simulation environment.

The selected car-like robot is a prototype built in the LAIR laboratory of the Mechanical Engineering School at the University of Campinas (Brazil) [22]. Figure 1 shows the real prototype and its 3D model.

2.1 Kinematics

The selected mobile robot is a car-type one, which is a widely used model for four-wheeled vehicles behavior analysis. The bicycle-like approach is an approximate model that eases the understanding and the modeling of any four-wheel car. This model has a front wheel for direction control and a back wheel for support.

The global quadrant corresponds to X_0, Y_0, and θ. The position of the car is represented by the frame X_1 and Y_1, where the X_1 axis points to the front of the car, aligned with the front and rear wheels. The vehicle's orientation and position are presented in Fig. 2 according to [23, 24].

The robot's kinematics equation regarding the local reference is described by the following Eq. (1), which follows the constant of Ackerman differential for the vehicle's direction regarding R.

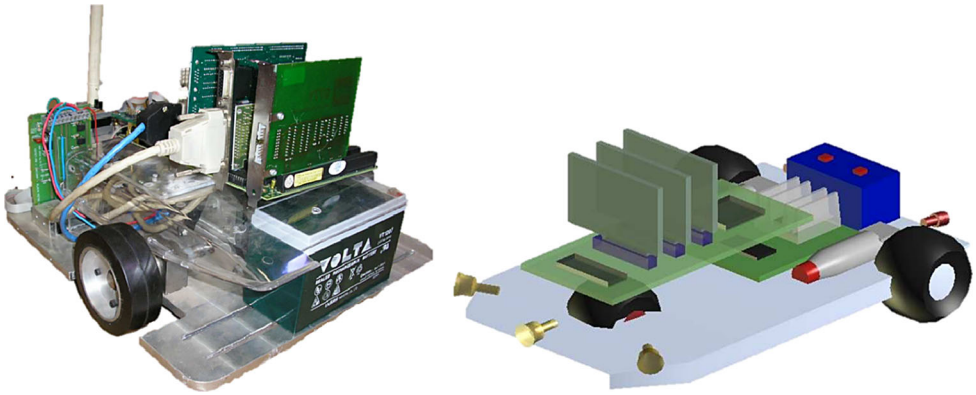


Fig. 1 Prototype and 3D model [22]

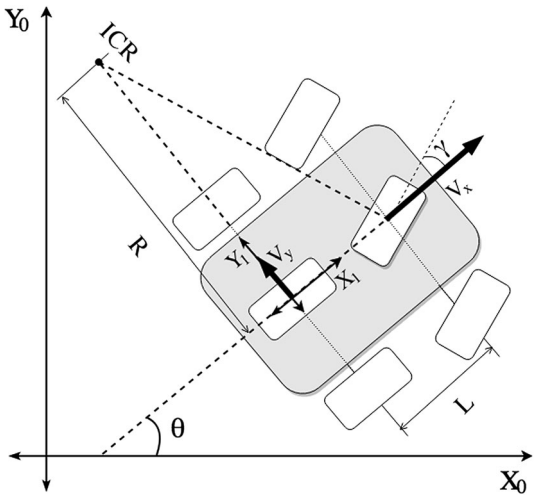


Fig. 2 Car-like robot kinematic model

$$V_x = v; \quad V_y = 0; \quad \dot{\theta} = \frac{v}{R} \quad \therefore \quad R = \frac{L}{\tan\gamma} \tag{1}$$

Subsequently, it is considered the global axis. The robot velocities seen from the global frame correspond to Eq. (2).

$$\dot{x} = v\cos\theta; \quad \dot{y} = v\sin\theta; \quad \dot{\theta} = \frac{v}{L}\tan\gamma \tag{2}$$

It is possible to obtain a kinematic restriction of the vehicle by Eq. (2), finding a relation between $\dot{x}$ and $\dot{y}$, using v. The obtained restriction is described by Eq. (3), showing a nonlinear relation and a dependency between $\dot{x}$ and $\dot{y}$, and the above is a constraint directly related to the trajectory that the car is able to follow.

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0 \tag{3}$$

The restriction described by Eq. (3) means that the robot cannot move if $v = 0$, which produces $\dot{\theta} = 0$. It is also important to indicate that γ cannot be equal to $\pi/2$; since being orthogonal to the rear wheel the robot cannot move, it indicates an undefined region [24].

In this case, the bicycle model approach used on the selected mobile robot requires the implementation of a mathematical relationship between velocity v and the input velocity for the right (v_r) and the left (v_l) wheels. There are two independent motors, one for each rear wheel, which is the reason for this requirement.

This mathematical relationship is based on the vehicle's instant center of rotation (ICR) and the steering wheel angle γ. The obtained expressions are given in Eqs. (4) and (5).

$$v_r = v\left(1 + \frac{\cos\theta}{2}\right) \tag{4}$$

$$v_l = v\left(1 - \frac{\cos\theta}{2}\right) \tag{5}$$

Using Eqs. (4) and (5), the robot velocity controller can be separated properly for both rear car wheels using the bicycle model system.

In addition, the defined constants used for the studied system are presented in Table 1.

Besides the defined sensors for the selected robot, sensors that allow a reactive navigation were selected as well, distance and compass sensors. The robot has 3 distance sensors arranged in front of its case and 2 on the back as shown in Fig. 1. Distance sensors are a group of analog IR-LED with a detection range of 4–30 cm; the sensor output is an analog voltage signal that keeps an inverse relation with the measured distance.

In addition, the location system for the training platform is based on the mobile robot kinematic model, according to the Eq. (2); those equations are numerically integrated using Euler's method to obtain the ideal vehicle position, as shown in Eq. (6) and the sampling time T is 1 ms in all

Table 1 Constants and mobile robot ranges

Constant	Value
L	0.2 m
γ	[− 30°, 30°]
v	[− 1, 1]

simulation cases. The hypothesis of the numerical integration can be made since the simulation environment is considered ideal, without slippage and where the robot wheels are in constant contact with the ground [24].

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x_0 + (T \cdot v \cdot \cos\theta) \\ y_0 + (T \cdot v \cdot \sin\theta) \\ \theta_0 + \left(T \cdot v \cdot \dfrac{\tan\gamma}{L}\right) \end{bmatrix} \tag{6}$$

The training platform based on the mobile robot kinematics allows an estimation of robot's position and orientation. It allows the mobile robot simulation as a system with inputs and outputs, which is perfect to test any kind of controller.

2.2 Dynamic approach

The dynamic model allows a clearer understanding of the robot behavior, considering the different forces interacting with the drive system. This mathematical approximation, based on physics, brings primal information to the drive system proper selection. In this case, the dynamic model for each robot wheel is the same, so only one model has to be developed. The dynamic model of the wheel is based on the Eq. (7).

$$\Gamma = J\ddot{\varphi} + B\dot{\varphi} + T_{\mathrm{p}} \tag{7}$$

where Γ is the motor torque interacting with the wheel dynamics, J is the inertia due to the angular wheel acceleration $\ddot{\varphi}$; B is the viscous friction coefficient due to the angular velocity $\dot{\varphi}$, and T_{p} is the perturbation torque. The value of J in this case is represented by the expression $J = \frac{1}{2}mr^2$; it corresponds to the inertia value of a cylinder on the Z axis, where m is the wheel mass expressed in kg and r is the wheel radius expressed in meters.

Also, T_{p} in this case corresponds to the wheel and ground interaction, which is represented by the Eq. (8), showing the relation between the gravity g, the wheel robot mass M, the wheel radius r and the static friction constant μ that is assumed as 1.

$$T_p = F_f r = mg\mu r \tag{8}$$

2.3 Transfer function definition and control implementation

Assuming that the drive system selection was made based on the described dynamics in Eqs. (7) and (8) and chosen the right way. The relation between the drive system and the external forces does not represent a big disturbance because the motors output torques are much higher than the resistance obtained by the dynamics values.

Following the methodology, the mathematical dynamics definition of each wheel and drive system is needed. To describe mathematically those relations, the model of a DC motor with load is used, as shown in the Eq. (9).

$$\frac{\varphi_L(s)}{E_a(s)} = \frac{K_t}{L_a J_{\mathrm{eq}} s^3 + \left(R_a J_{\mathrm{eq}} + L_a B_{\mathrm{eq}}\right)s^2 + \left(R_a B_{\mathrm{eq}} + K_b K_t\right)s} \tag{9}$$

where φ_L is the angular position of the wheel, E_a is the input voltage on the DC motor, K_t is the torque constant, L_a is the terminal inductance, J_{eq} is the equivalent inertia, R_a is the terminal resistance, B_{eq} is the equivalent viscous friction, K_b is the speed constant, J_{eq} and B_{eq} are the equivalent inertia and the equivalent system motor–gearbox–wheel viscous friction, respectively. The corresponding values are obtained as follows $J_{\mathrm{eq}} = J_m + J_L\left(\frac{N_1}{N_2}\right)^2$ and $B_{\mathrm{eq}} = B_m + B_L\left(\frac{N_1}{N_2}\right)^2$.

All the variables presented in the Eq. (8) are inner parameters of the DC motor, gearbox, and wheel; those parameters are presented in Table 2.

The graphical model described by the Eq. (9) that represents the DC motor load is presented in Fig. 3.

The obtained transfer function in the Eq. (9) should be controlled by reaching the desired wheel speed; for that purpose, in this case, the chosen controller is a PID tuned in Simulink® following the closed loop configuration as observed in Fig. 4.

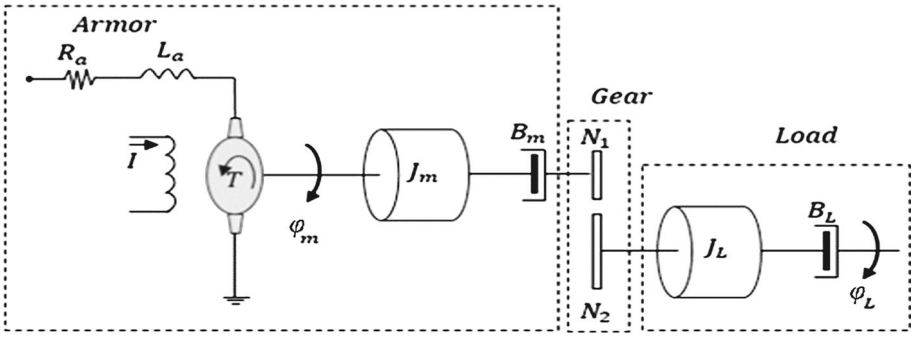
The PID controller function is shown in Eq. (10) and the tuned obtained values are presented in Table 3.

$$\mathrm{PID}(s) = P\left(1 + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}\right) \tag{10}$$

To implement the PID controller in a realistic scenario, it is important to analyze the discrete PID controller behavior too, since it can be embedded on a microcontroller to control the real prototype power of each wheel. The discrete PID has the structure presented in Fig. 5, with

Table 2 Gearbox and wheel system model motor parameters

Parameter	Value
R_a	0.212 Ω
L_a	0.077e−3 H
K_t	23.4e−3 N m/A
K_e	23.4e−3 V/(rad/s)
J_m	10.2e−6 kg m^2
B_m	2.57e−3 N m/rad
J_l	0.024 kg m^2
B_l	0.236 N m/rad
N_1	1
N_2	103
r	0.05 m



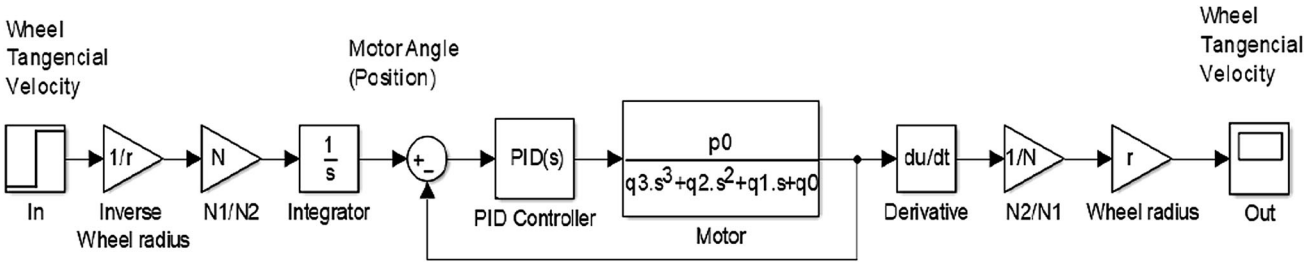


Fig. 3 Motor-load diagram

Fig. 4 PID motor control loop

Table 3 Tuned PID values

Parameter	Value
Proportional (*P*)	1.0140
Integral (*I*)	648.2393
Derivative (*D*)	0.00036
Filter coefficient (*N*)	1,241,053.46

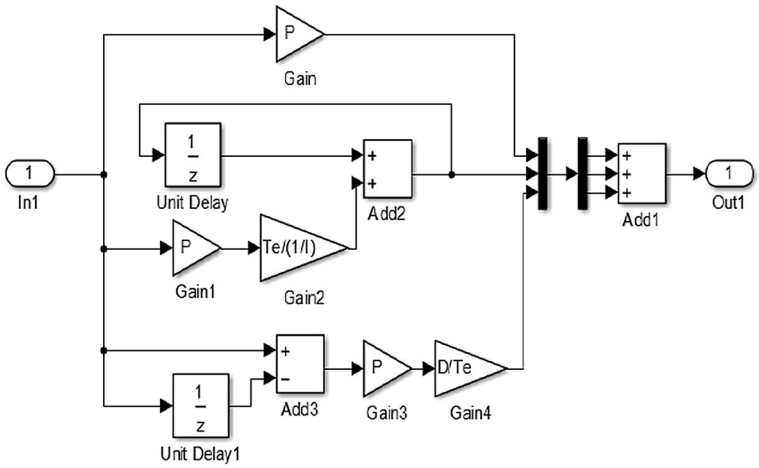
Table 4 Tuned discrete PID values

Parameter	Value
Proportional (*P*)	1.0140
Integral (*I*)	648.2393
Derivative (*D*)	0.00036
Sample time (Te)	0.1 ms

the parameters given in Table 4. The discrete PID structure was obtained using the backward finite difference method. The PID values of the continuous and the discrete controller are the same, as shown in Tables 3 and 4.

The steering wheel motor system is the same as the presented above, but in this case, the variable to control is the angular position instead of the angular velocity; the followed methodology is the same but adapted for the angular position case.

Fig. 5 Discrete PID structure



3 Neuroevolutive controller design

The current research is based on our previous work presented in [21]. It proposes a hybrid kinematic controller, based on the evolved neural networks with genetic algorithms to control a car kinematics to avoid obstacles. In contrast to that investigation, the current study includes the dynamics of the vehicle and a different approach for obstacle avoidance but based on the same controller principles.

According to the obtained results in [21], and after observing the obtained controller and the obstacle avoidance algorithm behavior, it was possible to conclude that the algorithms have many improvement spots, like choosing better inputs for the neural position controller or a better algorithm to avoid obstacles.

The current proposed neural controller has a single position controller. The initial reference of the position controller is the goal location, but this reference is modified when an obstacle appears. The position reference under the presence of an obstacle follows the value given by the potential fields algorithm [25], in order to avoid the obstacles detected. This potential field algorithm is calculated in the robot's local frame by using the mounted distance sensors. This algorithm will be named Local Potential Field (LPF), due to the current use of the algorithm.

3.1 Neuroevolutive position controller

To design the position controller, the NEAT algorithm was used; a detailed and specific explanation can be found in [2–4, 6, 26]. This was the selected method to create the neural network topology and its optimization by means of genetic algorithms, which allows an evolution of the neural network topology.

The NEAT algorithm was developed and implemented using Matlab® 2017a, and some modifications were made to make the algorithm behavior come closer to a memetic algorithm, according to [1]. The implemented algorithm is presented in Fig. 6.

The current proposal of a neuroevolutive position controller was based on [21] but adding some modifications due to the results of the realized tests. Those tests allowed to make some improvements to the controller inputs and outputs. The most significant improvement was the removal of a group of nonrelevant inputs, corresponding to the delayed outputs of the neurocontroller, parameters that did not affect the neurocontroller response. Due to the last consideration, the set of inputs correspond only to the error and delayed errors signals compared with the proposal of [21]. The proposed control loop diagram for the evaluation of the controller's fitness is presented in Fig. 7; it is possible to observe that the number of inputs of the selected neural controller is variable and corresponds only to the error of the system, and also that there are 2 outputs. Two different control loops were tested to verify and compare the simulation time and results, one of them with the dynamics and the other one without them.

As it is seen in Fig. 7, the implemented genetic algorithm fitness function which finally selects the best controllers for the system works as an optimizer criterion and regulator of the algorithm population. The chosen fitness function is presented in Eq. (11), where s represents the start of the steady-state controller response with a maximum total simulation time limit of 25% and n characterizes the simulation end. Also, ρ_o is the desired distance in polar coordinates and ρ is the current position in the same coordinate system, then $\rho_o - \rho$ is the instantaneous system error.

$$\text{Fitness}(k) = \frac{1}{n}\sum_{i=s}^{n}(\rho_o - \rho) + \left(\left(\frac{1}{n}\sum_{i=s}^{n}(\rho_o - \rho)\right)\left(\frac{\mathrm{d}(\rho_o - \rho)}{\mathrm{d}t}\right)\right) \quad (11)$$

3.2 Obstacle avoidance algorithm: potential field path planning

In contrast with the proposed algorithm in the last work [21], the current proposed method aims to have a mathematical and more exact approach to avoid the obstacles on the path, keeping the plasticity of the neural networks to adapt the position controller performance to the restrictions and the dynamics and kinematics of the system. In this case, the chosen method is the potential field path planning, which is used in a local frame by using the mounted robot sensors. Due to the current use of this algorithm and for keeping a logic name according to its purpose, it will be named Local Potential Field (LPF) path planning, where the used abbreviation will be LPF.

The original method of potential fields was proposed by Khatib [25] and was used in different cases for path planning in a local or global way, dynamic or static environments and different kinds of robots [27–29].

The path planning method creates a field, in this case around the robot and directs it to the desired position from a prior position. The potential field method treats a robot like a point under the influence of a potential field $U(q)$. The desired point represents an attractive force and the obstacles act as a repulsive force; the superposition of both forces guides the robot to the desired point while avoiding the obstacles [23]. The basic idea of the method proposes a differentiable potential field function $U(q)$ associated with a related artificial force $F(q)$, where q is a position $q = (x, y)$, as shown in Eq. (12).

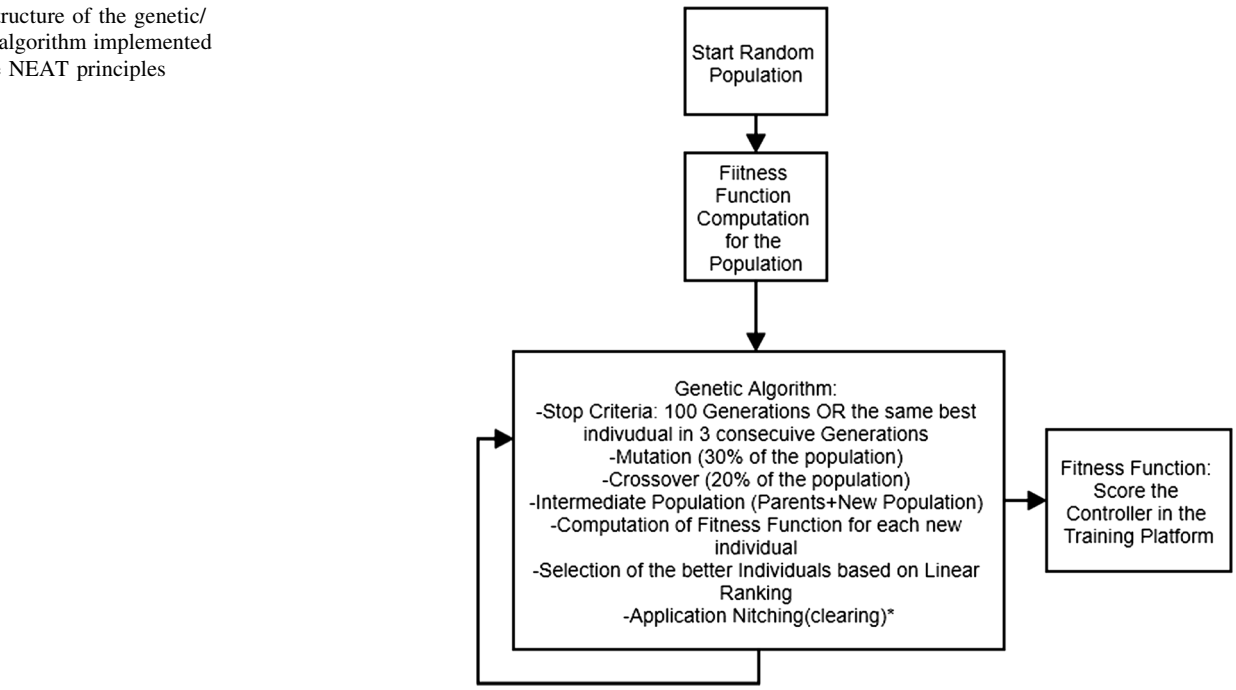


Fig. 6 Structure of the genetic/mimetic algorithm implemented using the NEAT principles

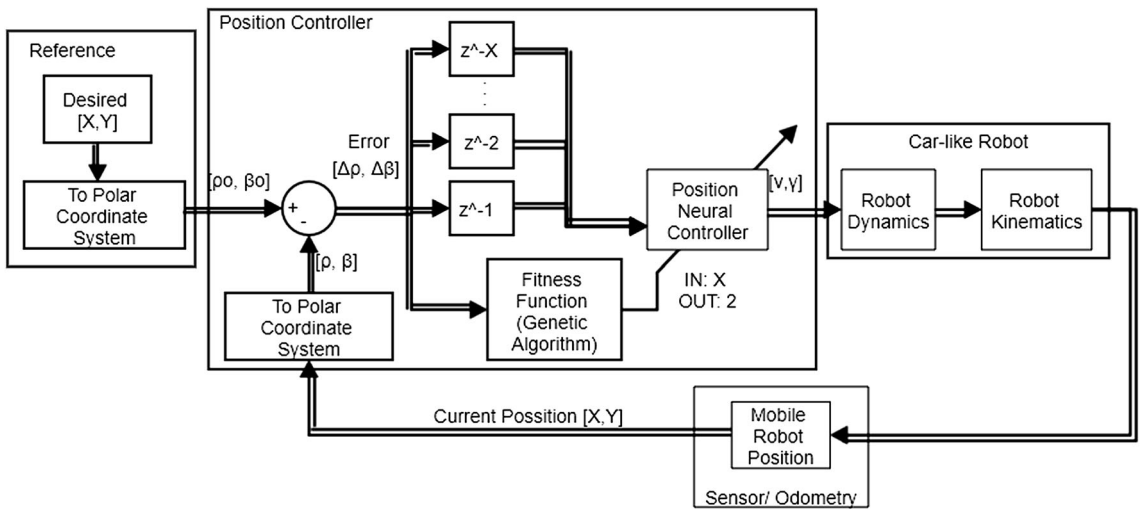


Fig. 7 Proposed control loop diagram for the neural position controller

$$F(q) = -\nabla U(q) \tag{12}$$

where $\nabla U(q)$ is the gradient of the vector U at position q, presented in the Eq. (13).

$$\nabla U = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix} \tag{13}$$

$U(q)$ acts as the superposition of the attracting and repulsive fields as is shown in the Eq. (14).

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \tag{14}$$

Consequently, it makes us think about the attractive and the repulsive potential fields, defined by the Eq. (15), where the attractive potential field can define a parabolic function.

$$U_{\text{att}}(q) = \frac{1}{2} k_{\text{att}} \rho_{\text{goal}}^2(q)$$
$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho(q) \le \rho_0 \\ 0 & \text{if } \rho(q) \le \rho_0 \end{cases} \tag{15}$$

where k_{att} and k_{rep} are positive scaling factors, $\rho_{\text{goal}}(q)$ corresponds to the Euclidean distance between

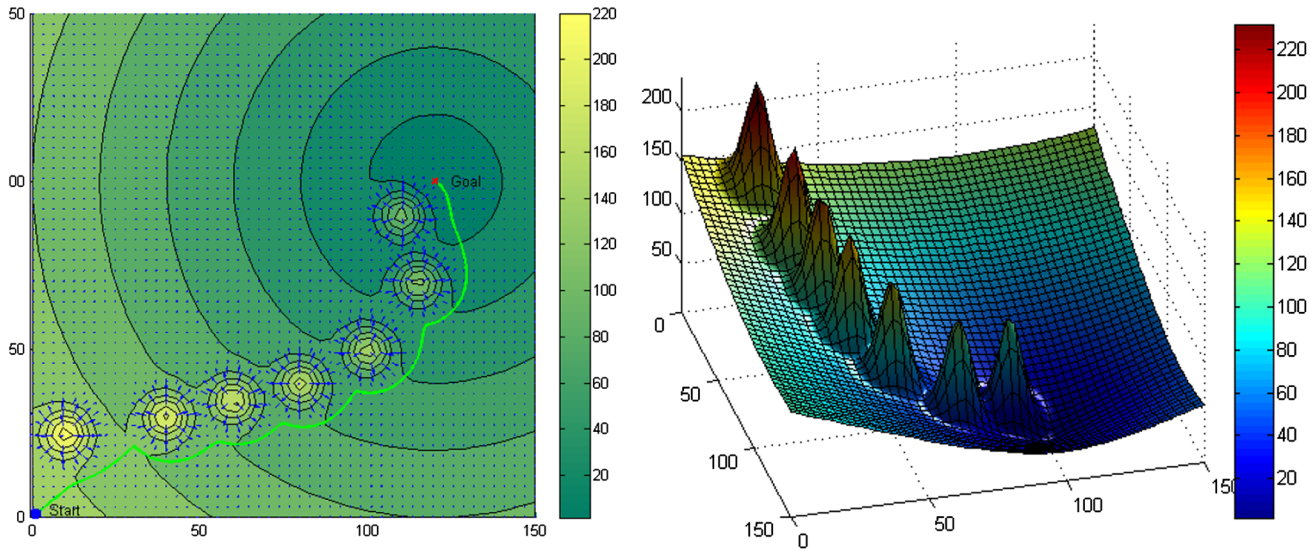


Fig. 8 Potential fields example

$||q - q_{\text{goal}}||$, $\rho(q)$ is the minimal distance from q to the object and ρ_0 is the distance of influence of the object.

The method works as Fig. 8 shows.

3.3 Position controller integration with the obstacle avoidance method

The integration of the position controller and the obstacle avoidance method is based on the reference changing that depends on an obstacle presence or absence. If an obstacle is sensed between the path of the car and the goal, the reference automatically is modified to follow the LPF method as the Fig. 9 shows.

It is possible to observe in Fig. 9 the same control structure proposed in the Fig. 7, the difference now is the variable reference that depends on the sensors state.

On one hand, the LPF algorithm target is to reach desired final position, so for the obstacle avoidance, the objective will be trying to achieve that target in a local way (sensor range). On the other hand, the change of reference from the desired final point to the given point by the LPF is activated when the sensors detect an obstacle on the path. The obstacle avoidance gives a reference to a point on the sensors ranges that will try to avoid the obstacles using the LFP method. An important point on the LPF method is the given goal to the robot, which is its desired final position.

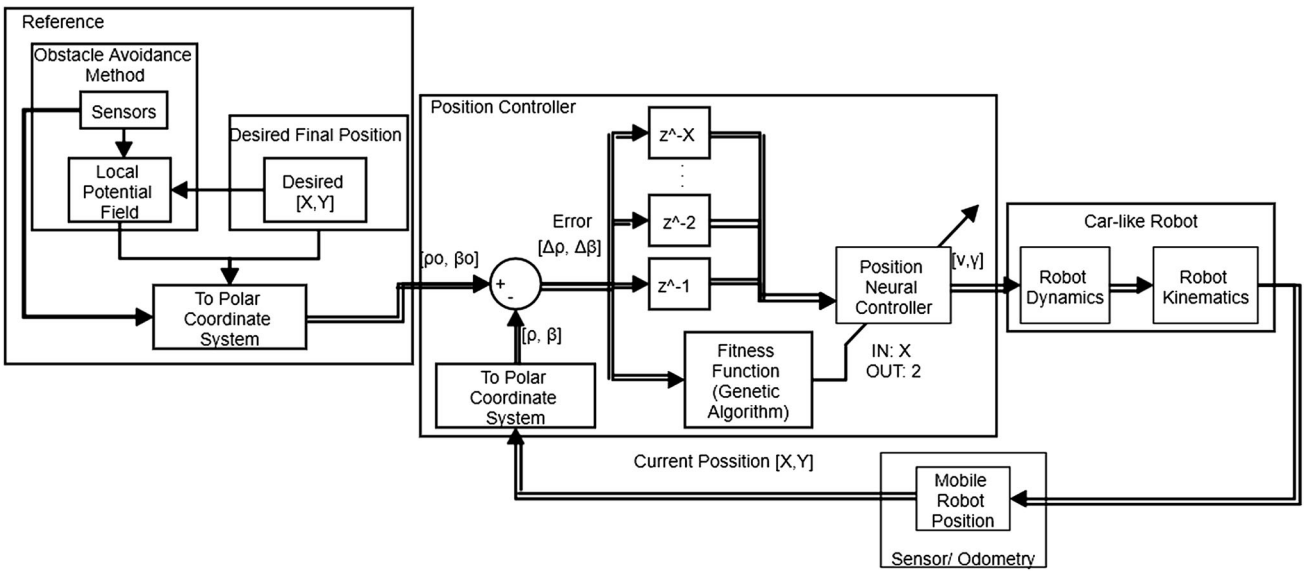


Fig. 9 Position controller integration with the obstacle avoidance method

4 Results and discussion

4.1 Position controller training results

For the controllers training and testing, the used computer was a Windows 10 laptop, with 4 GB of RAM, Intel Core i7 (2.4 GHz) processor, and NVIDIA GT 650M video card. Also, the used programming language was Matlab® 2017a, where the NEAT library was specifically implemented by the authors and the simulation environment included the robot physical behavior and the PID controllers.

For the position controller training, the fitness function means the lower value is better. All the tests were completed following the next conditions: initial vehicle position $(X, Y, \theta) = (0, 0, -\pi)$ and desired final position $(X, Y) = (1, 1)$, the same initial conditions presented by [21]. The controller inputs compared with the inputs presented by Caceres et al. [21] are quite different according to the explained in Sect. 3.1 and the Fig. 7.

The position controller was trained with and without the vehicle dynamics (mechanics), varying the number of inputs that corresponds to the delay error number. The objective of testing different training scenarios was the training time validation, the best fitness value and the number of generations needed to train the controller. The number of tests done was 10, starting in 1 delay until 15 delays, which represents the number of inputs (X) of the neural network position controller presented in the Figs. 7 and 9. Two different cases were tested: kinematic case and dynamic case (this also includes kinematic). The simulation results are summarized in Tables 5 and 6.

According to the listed results in Tables 5 and 6, it is possible to perceive that the best fitness value obtained in the kinematic case is always better than the dynamic case due to the complexity and response time. However, the best fitness value is easier to achieve using at least 3 delays. Results coincide with the basic discrete PID controller structure that uses exactly 3 delays. The results are presented in the Fig. 10.

Although the results listed in Tables 5 and 6 allow perceiving the training time of the kinematic and dynamic cases depending on the number of inputs, which permits to observe that the kinematic case training time is shorter than the dynamic training time. Another interesting point to observe is the almost linear relation between the number of inputs and the training time; this means that the more the inputs, the more training time is required. Results are shown in Fig. 11.

Another important thing would be the analysis of the average generations numbers needed for training a controller; this behavior is presented in Fig. 12.

Compared to the [21] work results, the training time was improved from an average of 6084 s to a lower value that depends on the input and training case as shown in Fig. 11. Also, the best fitness value obtained was 0.3092 and, in this case, the best fitness result obtained in the kinematic case was 0.3078 with 11 delays. The improvement of the control scheme results showed some advantages of using the algorithm.

Table 5 Kinematic case simulation results—10 test per each number of inputs

Number of inputs or delays	Training time (s)	STD training timE (s)	Average best fitness	STD average best fitness	Number of generations	STD number of generations	Best fitness of all the tests
1	400.93	286.34	0.94	0.56	8.40	3.89	0.3086
2	514.40	278.55	0.50	0.44	8.10	3.57	0.3088
3	801.77	473.72	0.33	0.06	11.20	4.54	0.3089
4	908.86	536.78	0.45	0.43	10.90	4.31	0.3090
5	938.27	644.99	0.31	0.01	11.00	7.44	0.3091
6	1139.86	1057.58	0.45	0.43	12.30	8.62	0.3093
7	1185.35	629.03	0.32	0.02	13.40	7.59	0.3094
8	1080.49	910.19	0.45	0.43	10.50	6.72	0.3095
9	1684.69	480.62	0.31	0.00	11.80	5.14	0.3096
10	2338.55	1189.70	0.31	0.01	13.10	4.04	0.3088
11	1674.49	1117.75	0.31	0.01	12.30	5.81	0.3078
12	1390.25	383.27	0.37	0.16	9.20	1.93	0.3104
13	1703.69	1225.34	0.59	0.57	10.70	7.02	0.3104
14	1393.77	600.91	0.31	0.00	9.60	3.57	0.3103
15	2019.21	852.04	0.45	0.42	12.20	4.49	0.3105

Table 6 Dynamic case simulation results—10 test per each number of inputs

Number of inputs or delays	Training time (s)	STD training time (s)	Average best fitness	STD average best fitness	Number of generations	STD number of generations	Best fitness of all the tests
1	2343.70	790.01	0.91	0.58	9.00	2.91	0.3869
2	2792.08	1147.80	0.55	0.34	9.80	3.91	0.3863
3	4104.15	1588.91	0.39	0.01	11.20	3.74	0.3864
4	3871.66	2037.03	0.42	0.11	9.10	4.43	0.3865
5	2703.37	1299.86	0.40	0.02	9.10	3.25	0.3869
6	3684.52	2115.60	0.47	0.26	11.30	6.31	0.3867
7	3405.01	1482.93	0.39	0.00	9.00	2.62	0.3869
8	3774.00	4293.21	0.39	0.01	9.40	3.81	0.3872
9	4125.50	4416.38	0.39	0.00	9.20	4.69	0.3874
10	4476.99	4539.56	0.39	0.01	12.80	5.65	0.3875
11	3580.48	3744.93	0.39	0.00	9.50	4.36	0.3878
12	5367.92	2950.30	0.41	0.05	8.90	3.54	0.3880
13	4772.87	4501.16	0.39	0.00	10.00	4.36	0.3909
14	6104.21	469.56	0.53	0.38	7.10	1.10	0.3887
15	6612.90	4377.52	0.39	0.01	14.11	9.47	0.3884

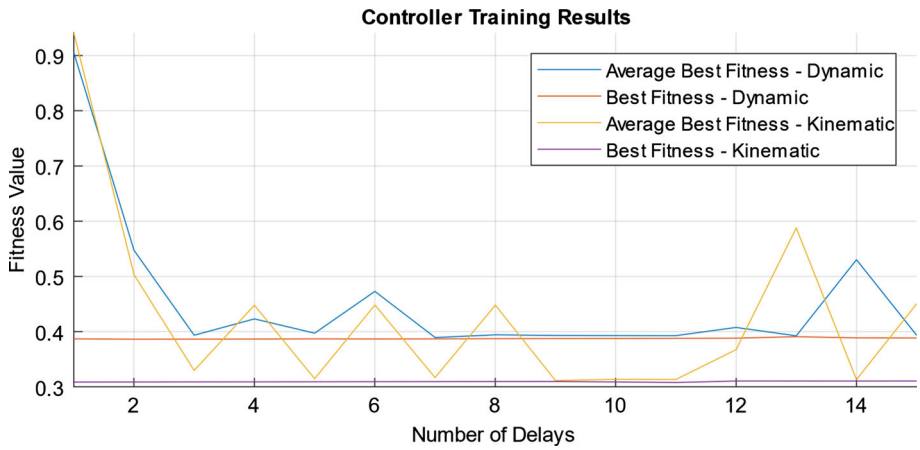


Fig. 10 Controller training results

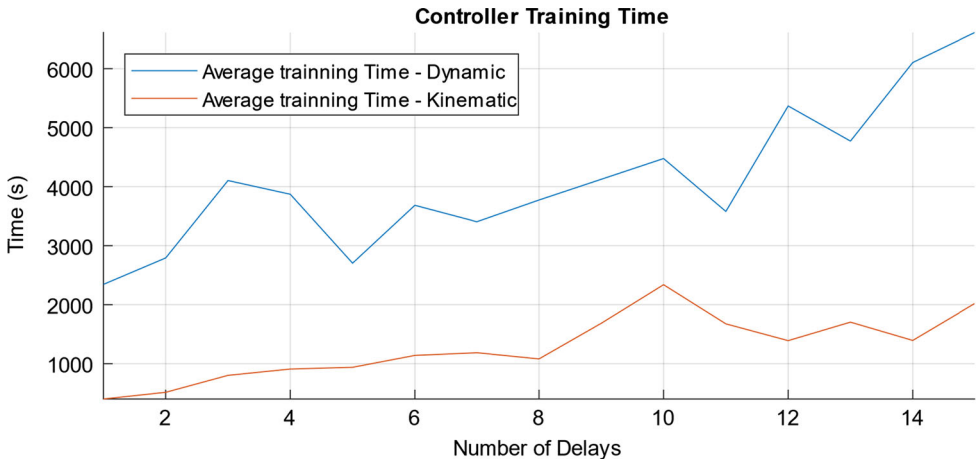


Fig. 11 Controller training time results

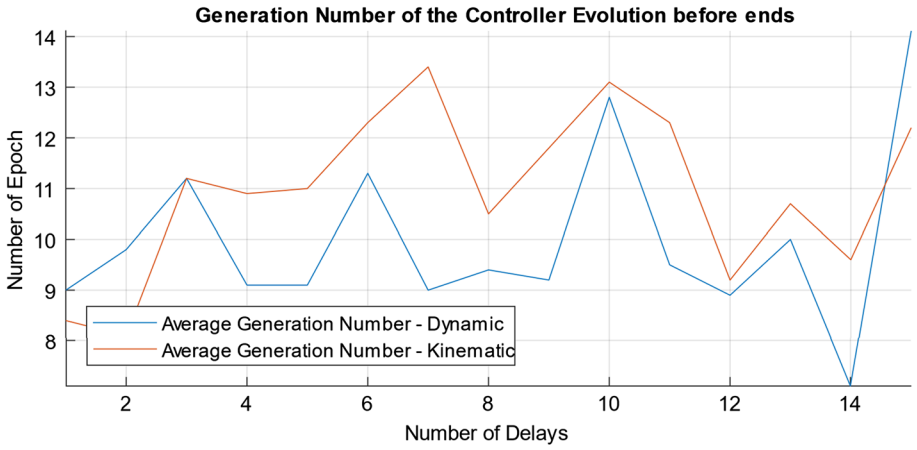


Fig. 12 Controller training average number of generations

4.2 Integration of the position controllers and the obstacle avoidance method

The integration of the position controller and the obstacle avoidance method following the Sect. 3.3 and the Fig. 9 was implemented. Some of the obtained controllers were tested. It is remarkable to remind the inclusion of the system dynamics and the kinematics for these tests and the similar results of the obtained controllers, making an approach to the real system.

Figure 13 presents a testing environment; it shows the path the vehicle took under the given conditions; the red dots are the obstacles, the green dot is the initial position, the yellow dot is the desired position, the scattered black line is the trajectory followed by the mobile robot and the black rectangle represents the mobile robot. Figures 14, 15, 16, 17, and 18 show different graphs of the controlled car detailed behavior using the proposed method for the analyzed conditions.

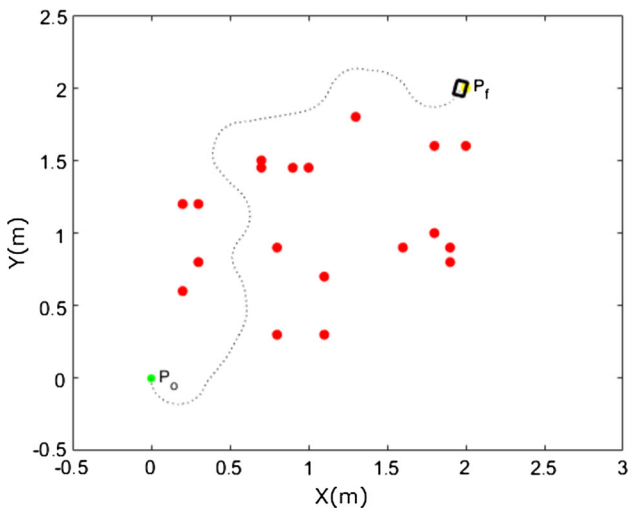


Fig. 13 Full control integration and response

Figures 17 and 18 allow perceiving the mechanical system dynamics. That dynamic behavior can be observed in the robot response (blue signal), trying to follow the signal of reference (red signal). The robot response has a delay whether it is compared with the signal of reference, due to the slower reaction time of an electromechanical system when compared with a computational system. The electromechanical response, being slower, corresponds to the mechanical robot dynamics given by the reaction time of the mechanical parts, like the DC motors and gearboxes. On the other hand, the computational response, which is faster, is the signal of reference given by the neuroevolutive controller.

4.3 The proposed method and the current state of the art

To find the position of the proposed method compared to similar works, it is essential to identify the differences of each implementation and the theoretical background. The current method uses a non-holonomic car-like robot that represents a constrained system; similar current works use holonomic robots like [11, 12, 18, 20, 30] that are non-constrained and fully controllable, a substantial difference to compare the proposed method. Other considerable difference is the physical (mechanical) analysis of the mobile robot, where most of the authors consider the mobile robot as a dot in the space [11, 12, 18, 20, 23, 24, 30] and others consider only the kinematics of the system [19]; in the current work, the systems considered the system kinematics and dynamics, including a PID controller for each wheel motor. Another theoretical consideration is the automatic control view, which holds an automatic control and mathematical theory that does not take into account the obstacle avoidance system, like the case of [31].

Under a similar technical approach, it is possible to compare the developed method with the method proposed by Caceres et al. [21]. In both cases, a similar position

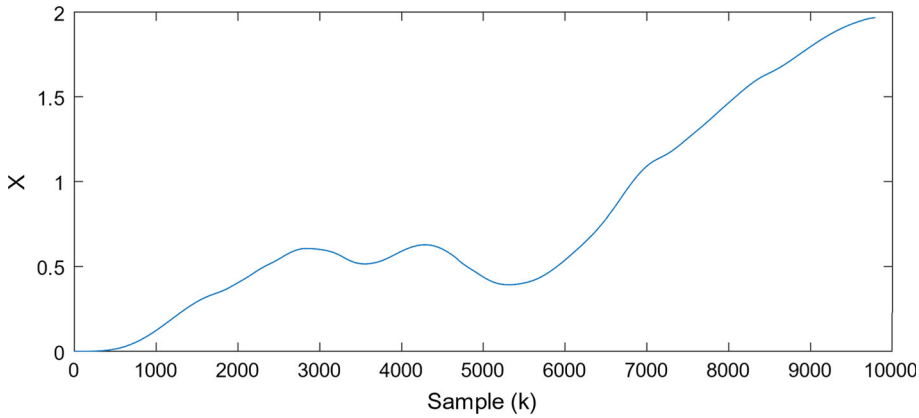


Fig. 14 Control response. *X*-axis

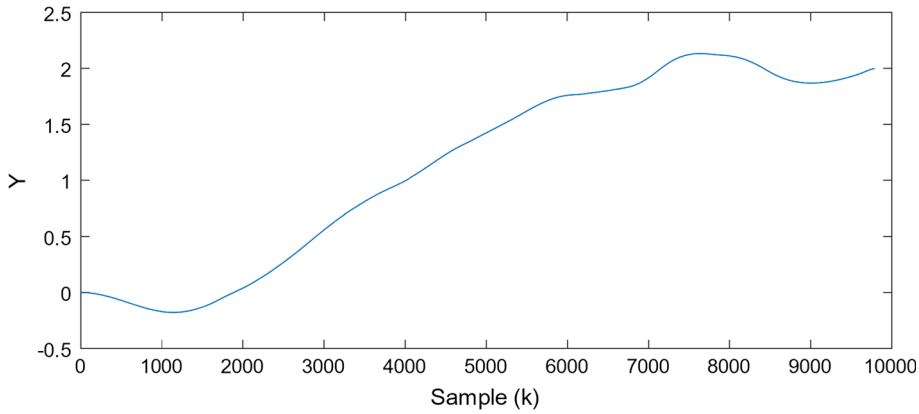


Fig. 15 Control response *Y*-axis

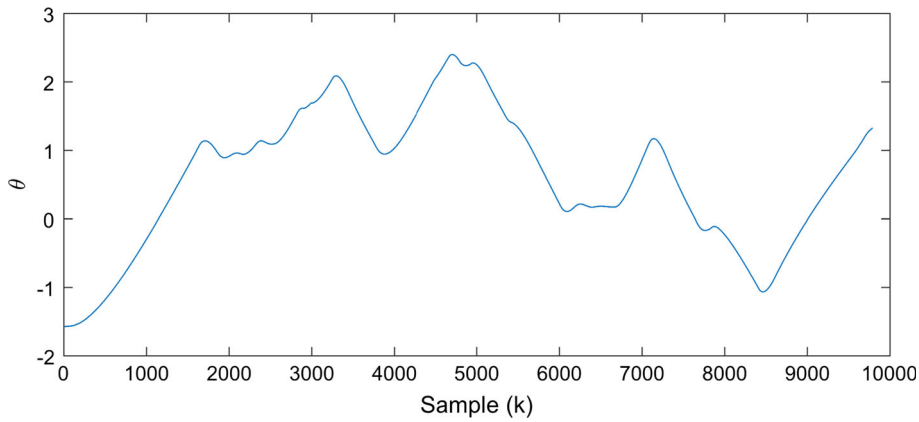


Fig. 16 Car angle control response

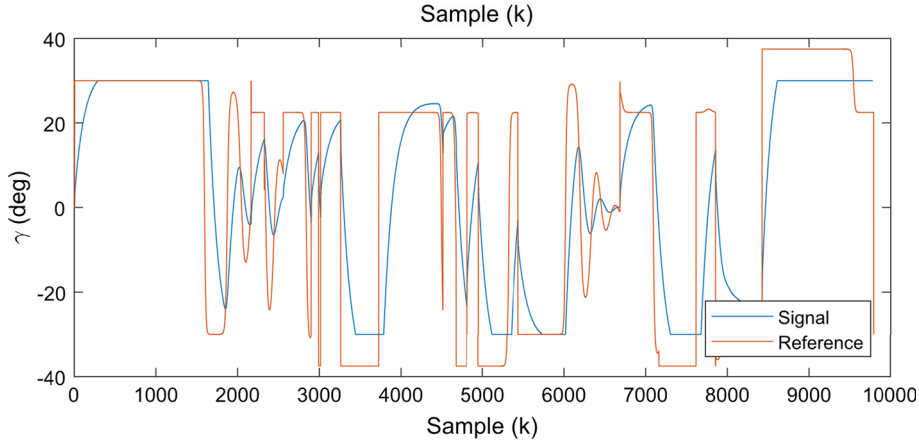


Fig. 17 Steering wheel signal

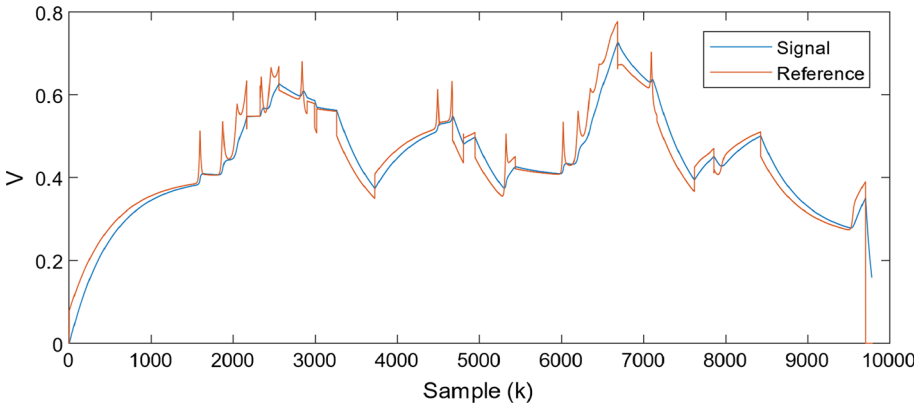


Fig. 18 Velocity signal

controller method was used in the same type of robot, using only its kinematics. The position controller method in both cases used the same fitness function and the same initial conditions, and the results showed an improvement in the fitness function result from 0.3092 in the work of [21] to 0.3078 in the current work (the lower the fitness, the better). The improvement was obtained due to the reduction in inputs to the neural network. Another improvement is solving an environment without crashing. The enhancement is related to the obstacle avoidance method and the improved response of the position controller; the value went from the 89 to a 94% in 100 simulated random environments.

The advantages and differences with the proposed method and similar works are the use of automatic control theory for the selection of the fitness function that allowed the natural selection to pick the more fit neural controllers. However, the neural network is used as a universal function for approximations. This means that it could be adapted to different conditions like dynamics and kinematics constraints, non-linearity functions and related.

5 Conclusions and further developments

From the current work results, it is possible to conclude that the neural controller designed using the NEAT algorithm reached the proposed objectives of controlling the kinematics and dynamics systems of a non-holonomic car-like robot in an unknown environment, avoiding obstacles and reaching the target point. Also, it was possible to determine that the kinematic controller works with the car's dynamics in a proper way under controlled conditions.

It is also important to highlight that the robot kinematic modeling was crucial because it is the basis of the training environment, which allows the controller to be tested and enables it to evolve differently. Additionally, the application of a simulation environment based on the electronic and mechanical characteristics of the robot allows the controller design process being faster and more efficient.

The selection of the fitness function based on the concepts of automatic control theory was also decisive to obtain a neural controller based on evolutionary algorithms. Although the usage of different fitness functions is an open topic to be studied in the future, with the study and comparison of a neural inverse controller and some other kinds of control strategies, the topic is an open issue in this study field.

The usage of the neural network as a universal controller approximation function must be emphasized, due to its adapting characteristics to different conditions. The adapting process of the neural network as a full robot controller is guided by the fitness function and the natural selection mechanism of the evolutionary system, obtaining a neuroevolutive controller as result.

Finally, an important aspect to remark is the influence of the kinematic and dynamic analysis of this system for the implementation and development of a soft computing algorithm to control a mobile robot, which is usually unnoticed and not considered in similar works.

Acknowledgements The authors wish to thank the University of Campinas and the National Council for Scientific and Technological Development (CNPq) for its support in the development of this work.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Sher GI (2013) Handbook of neuroevolution through Erlang. Springer, New York
2. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10:99–127
3. Siebel NT, Krause J, Sommer G (2007) Efficient learning of neural networks with evolutionary algorithms. In: Hamprecht FA, Schnörr C, Jähne B (eds) Pattern recognition. DAGM 2007.

Lecture Notes in Computer Science, vol 4713. Springer, Berlin, Heidelberg
4. Xu S, Moriguch H, Honiden S (2013) Sample efficiency analysis of neuroevolution algorithms on a quadruped robot. In: 2013 IEEE congress on evolutionary computation. IEEE, pp 2170–2177
5. Wen R, Guo Z, Zhao T, Ma X, Wang Q, Wu Z (2017) Neuroevolution of augmenting topologies based muscular-skeletal arm neurocontroller. In: 2017 IEEE international instrumentation and measurement technology conference (I2MTC). IEEE, pp 1–6
6. Raffe WL, Zambetta F, Li X (2013) Neuroevolution of content layout in the PCG: angry bots video game. In: 2013 IEEE congress on evolutionary computation. IEEE, pp 673–680
7. Rigatos GG (2011) Modelling and control for intelligent industrial systems. Springer, Berlin
8. Tellez A, Molina H, Villa L, Rubio E, Batyrshi I (2012) Parametric type-2 fuzzy logic systems. In: Fuzzy logic - algorithms, techniques and implementations. InTech
9. Jiménez MR, Cáceres C, Avilés O, Gordillo C (2012) Multi-tank fuzzy level controller system using system. 2012 IEEE ninth electronics, robotics and automotive mechanics conference (CERMA)
10. Castellano G, Castiello C, Fanelli AM, Jain L (2007) Advances in evolutionary computing for system design. Springer, Berlin
11. Aissa B, Fatima C, Yassine A (2017) Data fusion strategy for the navigation of a mobile robot in an unknown environment using fuzzy logic control. In: 2017 5th international conference on electrical engineering—Boumerdes (ICEE-B). IEEE, pp 1–6
12. Algabri M, Mathkour H, Ramdane H, Alsulaiman M (2015) Comparative study of soft computing techniques for mobile robot navigation in an unknown environment. Comput Hum Behav 50:42–56
13. Farooq U, Amar M, Asad MU, Hanif A, Saleh SO (2014) Design and implementation of neural network based controller for mobile robot navigation in unknown environments. Int J Comput Electr Eng 6:83–89
14. Mohanty PK, Parhi DR (2014) Navigation of autonomous mobile robot using adaptive network based fuzzy inference system. J Mech Sci Technol 28:2861–2868
15. Kim TS, Na JC, Kim KJ (2012) Optimization of an autonomous car controller using a self-adaptive evolutionary strategy. Int J Adv Robot Syst 9:1
16. Choset HM (2005) Principles of robot motion : theory, algorithms, and implementation. MIT Press, Cambridge
17. Pol RS, Murugan M (2015) A review on indoor human aware autonomous mobile robot navigation through a dynamic environment survey of different path planning algorithm and methods. In: 2015 international conference on industrial instrumentation and control (ICIC). IEEE, pp 1339–1344
18. Cao Z, Cheng L, Zhou C, Gu N, Wang X, Tan M (2015) Spiking neural network-based target tracking control for autonomous mobile robots. Neural Comput Appl 26:1839–1847
19. Rao AM, Ramji K, Sundara Siva Rao BSK, Vasu V, Puneeth C (2017) Navigation of non-holonomic mobile robot using neuro-fuzzy logic with integrated safe boundary algorithm. Int J Autom Comput 14:285–294
20. Chou C-Y, Juang C-F (2018) Navigation of an autonomous wheeled robot in unknown environments based on evolutionary fuzzy control. Inventions 3:3
21. Caceres C, Rosario JM, Amaya D (2017) Approach of kinematic control for a nonholonomic wheeled robot using artificial neural networks and genetic algorithms. In: 2017 international conference and workshop on bioinspired intelligence, pp 1–6
22. de Melo LF (2007) Proposta de simulador virtual para sistema de navegação de robos moveis utilizando conceitos de prototipagem rapida. http://repositorio.unicamp.br/handle/REPOSIP/265002?mode=full. Accessed 14 Dec 2017
23. Siegwart R, Nourbakhsh IR, Scaramuzza D (2011) Introduction to autonomous mobile robots, 2nd edn. MIT Press, Cambridge
24. Corke P (2011) Robotics, vision and control. Springer, Berlin
25. Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings. 1985 IEEE international conference on robotics and automation. Institute of Electrical and Electronics Engineers, pp 500–505
26. Caamano P, Bellas F, Duro RJ (2014) Augmenting the NEAT algorithm to improve its temporal processing capabilities. In: 2014 international joint conference on neural networks (IJCNN). IEEE, pp 1467–1473
27. Chiang H-T, Malone N, Lesser K, Oishi M, Tapia L (2015) Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In: 2015 ieee international conference on robotics and automation (ICRA). IEEE, pp 2347–2354
28. Ruchti J, Senkbeil R, Carroll J, Dickinson J, Holt J, Biaz S (2014) Unmanned aerial system collision avoidance using artificial potential fields. J Aerosp Inf Syst 11:140–144
29. Montiel O, Sepúlveda R, Orozco-Rosas U (2015) Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field. J Intell Robot Syst 79:237–257
30. Huang H-C (2016) Fusion of modified bat algorithm soft computing and dynamic model hard computing to online self-adaptive fuzzy control of autonomous mobile robots. IEEE Trans Ind Inform 12:972–979
31. Sun W, Tang S, Gao H, Zhao J (2016) Two time-scale tracking control of nonholonomic wheeled mobile robots. IEEE Trans Control Syst Technol 24:2059–2069