J Intell Robot Syst (2009) 56:469–484
DOI 10.1007/s10846-009-9332-z

Evolution of Fuzzy Controllers for Robotic Vehicles: The Role of Fitness Function Selection

L. Doitsidis · N. C. Tsourveloudis · S. Piperidis

Received: 19 February 2009 / Accepted: 7 April 2009 / Published online: 23 April 2009

Abstract An important issue not addressed in the literature, is related to the selection of the fitness function parameters which are used in the evolution process of fuzzy logic controllers for mobile robot navigation. The majority of the fitness functions used for controllers evolution are empirically selected and (most of times) task specified. This results to controllers which heavily depend on fitness function selection. In this paper we compare three major different types of fitness functions and how they affect the navigation performance of a fuzzy logic controlled real robot. Genetic algorithms are employed to evolve the membership functions of these controllers. Further, an efficiency measure is introduced for the systematic analysis and benchmarking of overall performance. This measure takes into account important performance results of the robot during experimentation, such as the final distance from target, the time needed to reach its final position, the time of sensor activation, the mean linear velocity e.t.c. In order to examine the validity of our approach a low cost mobile robot has been developed, which is used as a testbed.

Keywords Mobile robots · Evolutionary robotics · Fuzzy logic

1 Introduction

Fuzzy logic techniques are commonly used for navigation of different types of robot vehicles [1]. The popularity of fuzzy logic is based on the fact that it can cope with the uncertainty of the sensors and the environment really well. By using it, the robotic

L. Doitsidis (✉) · N. C. Tsourveloudis · S. Piperidis
Intelligent Systems and Robotics Laboratory, Department of Production Engineering and Management, Technical University of Crete, University Campus, Chania 73100, Greece
e-mail: ldoitsidis@dpem.tuc.gr

vehicles are able to move in known or unknown environments, using control laws that derive from a fuzzy rule base. This base is consisted from a set of predefined IF–THEN rules, which remains constant during the operation of the robot. These rules along with the membership functions of the fuzzy variables are usually designed ad hoc by human experts.

Several researchers have used fuzzy logic for the navigation of mobile robots. In [2], a layer goal oriented motion planning strategy using fuzzy logic controllers has been proposed, which uses sub-goals in order to move in a specific target point. Another approach is presented in [3], where the authors propose a control system consisted of fuzzy behaviors for the control of an indoor mobile robot. All the behaviors are implemented as Mamdani fuzzy controllers except one which is implemented as adaptive neuro-fuzzy. In [4] a combined approach of fuzzy and electrostatic potential fields is presented that assures navigation and obstacle avoidance. The main drawback of these approaches is that the design of the fuzzy controllers is relied mainly on the experience of the designer. In order to overcome this problem several researchers have proposed tuning of the fuzzy logic controller based on learning methods [5] and evolutionary algorithms [6–11], in an attempt to improve the performance and the behavior of the control procedure.

In [6], a fuzzy logic controller for a *khepera* robot in a simulated environment was evolved using a genetic algorithm, and the behaviors of the evolved controller were analyzed with a state transition diagram. The robot produces emergent behaviors by the interaction of the fuzzy rules that came out from the evolution process. In [7], the authors proposed a three step evolution process to self-organize a fuzzy logic controller. The procedure initially tunes the output term set and rule base, then the input membership functions, and in the third phase it tunes the output membership functions. Hargas et al. in [8], proposed a fuzzy-genetic technique for the on-line learning and adaptation of an intelligent robotic vehicle. In [9] the authors present a methodology for tuning the knowledge base of the fuzzy logic controller based on a compact scheme for the genetic representation of the fuzzy rule base. In [10] the authors present a scheme for the evolution of the rule base of a fuzzy logic controller. The evolution takes place in simulated robots and the evolved controllers are tested on a *khepera* mobile robot. Nanayakkara et al. in [11], present an evolutionary learning methodology using a multi objective fitness function that incorporates several linguistic features. The methodology is compared to the results derived from a conventional evolutionary algorithm. An attempt to formulate a way of picking the suitable function for a task was made by Nolfi and Floreano in [12]. They proposed the concept of "fitness space", which provides a framework for describing and designing fitness functions for autonomous systems.

An important issue not addressed in the literature, is related to the selection of the fitness function parameters which are used in the evolution process of fuzzy logic controllers. The majority of the fitness functions used for controllers evolution are empirically selected and (most of times) task specified. This results to controllers which heavily depend on fitness function selection.

In this paper we compare three major different types of fitness functions based on a classification analysis introduced in [13]. This comparison is based on the navigation performance of a fuzzy logic controlled real robot. The different kinds of fuzzy controllers (produced by each fitness function) are used and analyzed. Genetic algorithms are employed to evolve the membership functions of these controllers.

Further, an efficiency measure is introduced for the systematic analysis and benchmarking of overall performance. This measure takes into account important performance results of the robot during experimentation, such as the final distance from target, the time needed to reach its final position, the time of sensor activation, the mean linear velocity e.t.c.

The paper is organized as follows. In Section 2 a brief description of different fitness function categories and their main characteristics are presented. In Section 3 the fuzzy logic controller which is used in the evolution process is presented together with the genetic algorithm. The three different types of fitness functions considered in this study are identified and formally analyzed. In Section 4 a custom made robotic vehicle used for experimentation is presented and the hardware and software components are described in detail. In Section 5 experimental results are presented and analyzed. Finally in Section 6, issues for discussion and further research are presented.

2 Fitness Function Categories

The choice of the fitness function is a fundamental issue for the evolution of the controller of mobile robots. In [13] Nelson et al. presented a detailed analysis and classification of the different types of fitness functions used in evolutionary robotics (ER). The fitness function is at the heart of an evolutionary computing application. It is responsible for determining which solutions within a population are better at solving the particular problem at hand. In work attempting to evolve autonomous robot controllers capable of performing complex tasks, the fitness function is often the limiting factor in controller's quality. This limit is usually manifested by a plateau in fitness evaluation in later generations, and indicates that the fitness function is no longer able to detect differences between individuals in the evolving population. In [13], a classification hierarchy based on the degree of a priori knowledge that is reflected in the fitness functions used to evolve behaviors or task performance abilities is presented.

The justification for using a priori knowledge as a basis for classification and organization of the research is that it reflects the level of truly novel learning that has been accomplished [14]. There are of course other means by which designers introduce their own a priori knowledge of task solutions into the design of experimental systems intended to study evolution (or learning) in autonomous robots. These include selection of appropriate sensors and actuators, design of training environments, and choice of initial conditions. Although these other forms of introduced a prior knowledge are also important, it is the fitness function that contains the most explicit and varied forms of task solutions knowledge. Many of the research platforms have at least qualitative commonalities of sensor capabilities and actuator arrangements. Seven broad classes of fitness functions are presented in Table 1, as they are identified and described in detail in [13].

Based on this categorization we considered *behavioral*, *aggregate* and *tailored fitness functions* for the evolution of the fuzzy logic controller of a mobile robot. The selection of these types was based on the fact that the majority of the fitness functions used in the literature is belonging to these categories [13].

Table 1 Fitness function classes

Fitness function class	A priori knowledge incorporated
Training data fitness functions (for use with training data sets)	Very high
Behavioral fitness functions	High
Functional incremental fitness functions	Moderate-high
Tailored fitness functions	Moderate
Environmental incremental fitness functions	Moderate
Competitive and co-competitive selection	Very low–moderate
Aggregate fitness functions	Very low

The term behavioral fitness, describes task-specified hand-formulated functions that measure various aspects of a robot's functionality. The distinctive feature of this class of functions is that they are made only of terms or components that select for behavioral features of a presupposed solution to a given task. For example, if one wished to evolve robots to move about an environment and avoid obstacles, one might include a term in the fitness selection function that is maximized if a robot turns when its forward sensors are stimulated at close range. In this case the system is set up such that robots will evolve to produce a certain actuator output in response to a given sensor input.

Aggregate fitness functions select only for high-level success or failure to complete a task without regard to how the task was completed. This type of selection reduces injection of human bias into the evolving system by aggregating the evaluation of benefit (or deficit) of all of the robot's behaviors into a single success/failure term. Consider the following foraging task: a robot is to locate and collect objects and then deposit them at a particular location (or a "nest"). An aggregate fitness function would contain information only related to task completion. Suppose the task is considered to be complete when an object is deposited at the nest. An example of an aggregate fitness function for this task would be one that counted the number of objects at the nest after the end of a trial period.

Finally, the tailored fitness functions are responsible for the evolution of behaviors that are already well known by the designer of the evolution process. This type of functions combines elements from both the behavioral and the aggregate categories. Usually in tailored functions, the aggregate terms are measuring the completion of partial task in a way that invokes some degree of a priori knowledge. As an example, suppose a phototaxis behavior is to be evolved. A possible fitness function might contain a term that rewards a controller that arrives at the light source by any means, regardless of the specific sensor-actuator behaviors used to perform the task. This term would be considered an aggregate term. If it were the only term in the fitness function, then the whole function would be considered aggregate. If the function also contained a second behavioral term, for example, one that maximized the amount of time the robot spent pointing toward the light source, then the two terms together would constitute an example of a tailored fitness function. Note that this second term, selecting for pointing toward the light source, does represent implicit assumptions about the structure of the environment and may not be the best way to approach the light source in some complex environments.

3 Fuzzy Logic Controller and the Evolution Process

3.1 Fuzzy Logic Controller Design

On previous work [15, 16] we have presented modular fuzzy logic controllers which were able to navigate mobile robots in unknown environments, based on sensor's feedback. Based on this work we have developed a similar controller adapted to the testbed which is going to be described.

The controller has four inputs and two outputs. The inputs are the *heading error*, the *distance from obstacles* as it is measured from the front sonar sensor and *the distance from obstacles* as it is measured from the left and the right infrared sensors. The outputs from the controller are the speeds of the left and right servo motor. The inputs from the sensors are used in order to calculate the collision possibilities in three directions: front, left and right. The *heading error* is calculated from the robot's current heading and the desired heading.

Implementation wise the input variable heading error includes four trapezoidal membership functions and one triangular membership function. The input variable *front collision possibility* includes three trapezoidal membership functions and the input variables *left* and *right collision possibility* are including two trapezoidal membership functions each.

The value of each distance input variable d_i $i = 1,\ldots,3$ (corresponding to *Front*, *Left*, *Right Area*) are expressed by the fuzzy sets C_i, A_i (corresponding to *Close* and*Away*) for the left and right area and C_i, MD_i, A_i (corresponding to *Close*, *Medium Distance* and *Away*) for the front area. The values of the input variable *heading error*, (he), are expressed by the fuzzy sets *FL, L, AH, R, FR* (which are *Far Left, Left, Ahead, Right and Far Right*). The outputs of the fuzzy logic controller are the speeds of the left and the right servo motor. The membership functions describing the fuzzy sets for each output variable are Z, S, M, H (which are *Zero, Small, Medium and High Speed*).

Each fuzzy rule j is expressed as:

$$\begin{aligned}&IF\ he\ is\ HE_j\ AND\ d_1\ is\ D_{j1}\ AND\ d_2\ is\ D_{j2}\ AND\ d_3\ is\ D_{j3}\\&\quad THEN\ lm\ is\ LM_j\ AND\ rm\ is\ RM_j\end{aligned}$$

where $j = 1,\ldots,$ number_of_rules, D_{ji} is the fuzzy set for d_i in the jth rule which takes the linguistic value of C_i, MD_i, A_i for $i = 1$ (front area) and C_i, A_i for $i = 2,3$ corresponding to the left and right areas. HE_j takes the linguistic values *FL, L, AH, R, FR* and LM_j and RM_j are taking the linguistic values *Z, S, M, H*.

The generic mathematical expression of the jth navigation rule is given by:

$$\mu_{R^{(j)}}(he, d_i, lm, rm) = \min\left[\mu_{HE^{(j)}}(he), \mu_{D_i^j}(d_i), \mu_{LM^{(j)}}(lm), \mu_{RM^{(j)}}(rm)\right]. \quad (1)$$

The overall navigation output is given by the max-min composition and in particular:

$$\mu_N^* (lm,\ rm) = \max \min_{he,\ d_i} \left[\mu_{AND}^* (he,\ d_i)\,,\ \mu_R (he,\ d_i,\ lm,\ rm)\right], \tag{2}$$

where, $\mu_R (he,\ d_i,\ lm,\ rm) = \bigcup_{j=1}^{J} \mu_{R^{(i)}} (he,\ d_i,\ lm,\ rm)$.

3.2 Genetic Fuzzy Tuning

The membership functions and the rule base of a fuzzy controller are usually the potentional candidates for evolution [17]. In this approach we consider a fixed rule base, as it was created by the human expert that designed the fuzzy logic controller, and we evolve the membership functions by using a genetic algorithm.

The chromosome is created by encoding the real values of the numbers that define the membership functions of the input and output variables, as described in Fig. 1. The same approach has been used successfully for the evolution of fuzzy logic controllers for different type of applications i.e. manufacturing systems [18].

Each input and output variable is encoded as an array Cin_i where $i = 1, \ldots,$ number_of_input_variables, and each output variable is encoded as $Cout_j$, where $j = 1, \ldots,$ number_of_output_variables. The overall chromosome has the following form:

$$C = \left[C\text{in}_i\ C\text{out}_j\right] \tag{3}$$

The length of chromosome is related to the number and type of the input and output membership functions. An initial population is created and each individual, which is a fuzzy logic controller, of a single generation is tested in the same experiment. After the completion of the experiments the performance of each individual is evaluated based on the fitness function used, and the individuals are ranked. For the selection process, a roulette wheel with slots according to fitness is used, as described in [19]. After that the crossover is performed and mutation of the final population.

In order to investigate the influence of different types of fitness functions to the evolution of fuzzy logic controllers for the navigation of a mobile robot, three different fitness functions were selected based on the classification presented at Section 2. The task of the robotic vehicle was to start from a given known position and navigate to a target position in an unknown environment. During experimentation the crossover, the mutation probability and all other parameters of the genetic

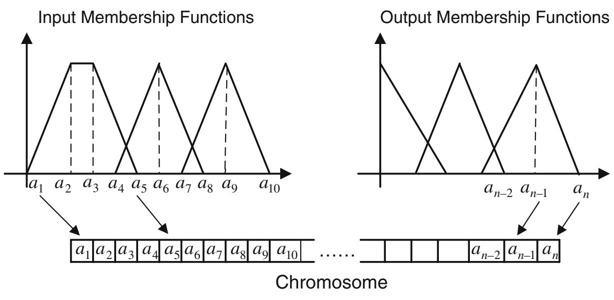
Input Membership Functions
Output Membership Functions
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
a_{n-2} a_{n-1} a_n
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} ...... a_{n-2} a_{n-1} a_n
Chromosome

Fig. 1 Chromosome created by the membership functions

algorithm remained the same. As a result the structure of the evolved controllers in all cases is attributed to the fitness function used.

The first function used belongs in the category of the *behavioral* fitness functions similar to the one used in [20]. The fitness was considering the percentage of straight motion of the vehicle, the activation level of the sensors and the velocity in each wheel of the vehicle. This function does not consider the distance of robot from the target position. The form of the function for the individual i of the generation j is:

$$f_i = \text{mean}\,(v_\text{L}, v_\text{R})\,(a_\text{F} + a_\text{L} + a_\text{R}) \tag{4}$$

where, v_L, v_R are the velocities of the left and the right wheel of the robot respectively, and $\alpha_\text{F}, \alpha_\text{R}, \alpha_\text{L}$ are the activation levels of the front, right and left sensors. The term "activation level" is given by the following equation:

$$a_\text{sensor} = \frac{\text{sensor\_reading }(t)}{\text{max sensor range}}, \tag{5}$$

where, sensor_reading (t) is the reading at the time step t of the experiment. Activation level equal to 1, means that no obstacle is detected by this sensor.

The second fitness function used, was a *tailored* one, which measures how close the robot has went in a target position comparing to its initial position and the activation level of each sensor that the robot had. The form of the function for the individual i of the generation j is:

$$f_i = \left(\left(3e^{2\frac{d_\text{final} - d_\text{initial}}{d_\text{initial}}}\right)/\text{ct}\right) + (a_\text{F} + a_\text{L} + a_\text{R}), \tag{6}$$

where, d_final is the final distance of the robot from a predefined target point, d_initial the initial distance of the robot from a predefined target point, and ct is a constant used.

The third fitness function used, belongs in the category of the *aggregate* functions and measures only how close the robot has gone towards in a target position comparing to its initial position. During the evolution process the level of activation of the sensors or the number of collisions weren't considered. The fitness function for the individual i of the generation j is:

$$f_i = \left(3e^{2\left(\frac{d_\text{final} - d_\text{initial}}{d_\text{initial}}\right)}\right)/\text{ct}. \tag{7}$$

4 HELOT Custom Robotic Vehicle

There is a lot of argument about using simulation or experiments in real robots for the evolution of their controllers. Some researchers propose the use of simulators to initially evolve the controllers, and then validate them in real robots. We believe that although simulation has proven to be a useful tool for the evolution of robot controllers, the evolution in real robots can incorporate factors that a simulation, no matter how accurate it is, cannot consider. For this reason we have designed and developed a low cost robot that allows experimentation and validation of our approach. All the work presented in this manuscript has been done on a real robot including the evolution process. We will briefly describe the robot's parts together with the essential software developed for control and sensor processing.

4.1 Hardware Design

The overall design of the vehicle was made based on the philosophy of cost reduction without minimizing the capabilities of the vehicle. A low budget commercial mobile platform was chosen to be the base for the robot evolution and testing of different devices. Several modifications were made and new sensors and devices were added since the minimal configuration of the initial platform wasn't suitable for experimentation. The HELOT vehicle is shown in Fig. 2. The sensor suite of the robot is consisted of a sonar range device positioned in the center of the upper front part, and two infrared sensors positioned in the lower left and right part of the vehicle. It has two odometers one in each wheel and an electronic compass in the upper center part. The two wheels of the robot are driven from one servo motor each. The low level control of the servo motors and the data acquisition from the sensors and the other devices is performed from a board equipped with an OOPic microprocessor. A Bluetooth device is mounted on HELOT, which allows the communication of the robot with a host computer also equipped with Bluetooth.

The experimentation area with a HELOT robot inside is presented in Fig. 3. All experiments were performed in this 3.7 × 4.5 m arena. The floor is covered with a carpet so as to minimize robot's sliding, which might lead to erroneous odometer readings.

4.2 Software Design

An efficient software design is crucial for optimizing the functionality of the robotic research platform. With the microprocessor onboard the robotic vehicle has limited capabilities and it can't accommodate sophisticated algorithms for control and decision making. To overcome this problem a host/slave system consisted of a base station and the robot was used. The base station is responsible for the execution of complicated algorithms while the microcontroller onboard the robot

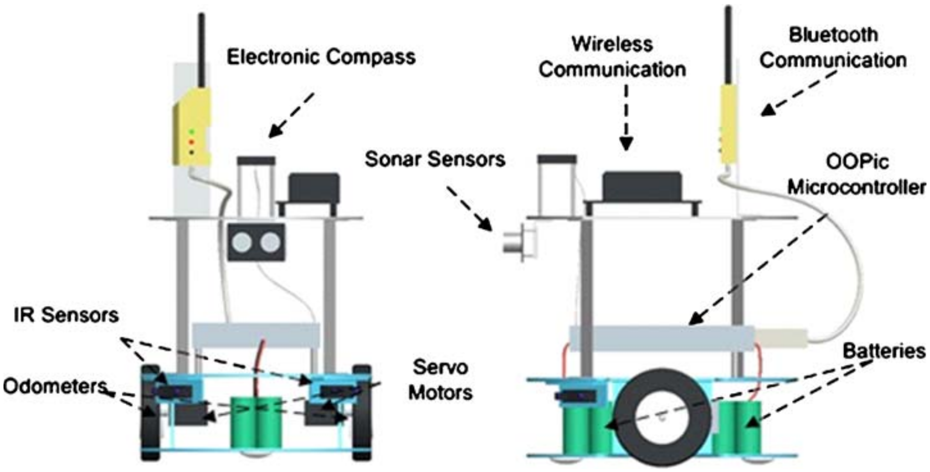

Fig. 2 HELOT's sensors configuration

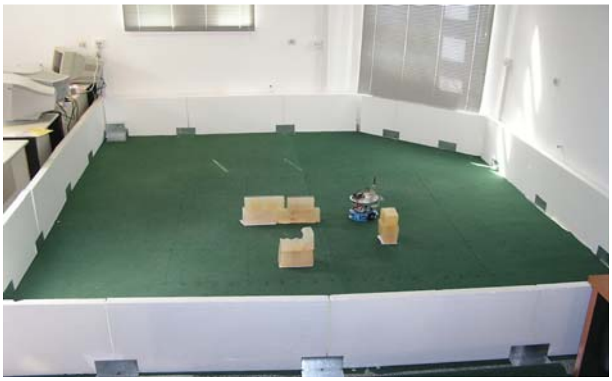
Fig. 3 Experimental testbed

is managing the devices and actuators and feeds unprocessed data from the sensors to the remote computer via a wireless RS232 link accommodated by a Bluetooth device. It should be noted that although the configuration is a host/slave system, conceptually the robot is considered as an autonomous agent. The software for device management running on the robot was developed in C, while the software running on the base station was entirely developed in Matlab. The advantages of this type of implementation were analyzed in [21]. The relationship of the software running on the base station with the software running onboard the robot is illustrated in Fig. 4.

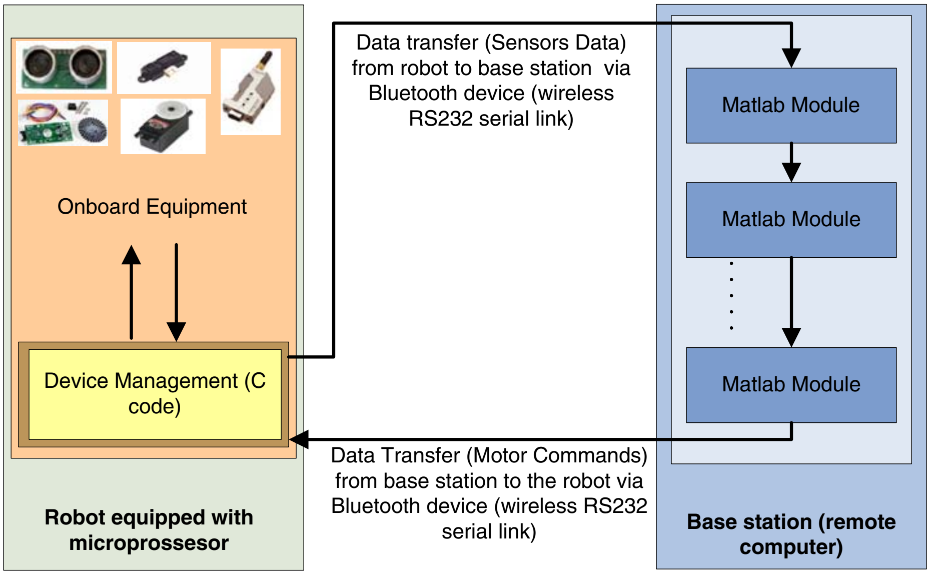
Data transfer (Sensors Data) from robot to base station via Bluetooth device (wireless RS232 serial link)
Onboard Equipment
Device Management (C code)
Matlab Module
Matlab Module
Matlab Module
Data Transfer (Motor Commands) from base station to the robot via Bluetooth device (wireless RS232 serial link)
Robot equipped with microprossesor
Base station (remote computer)

Fig. 4 Data flow diagram illustrating the relationship of the software running on the base station (high level control) and the robot (low level control)

Different Matlab modules were developed to process the data from the sensors. The data from the odometer were used and fed to the model described in [22] for differential drive robots in order to evaluate the robot's position.

5 Experimental Results and Analysis

For each fitness function type, we evolved the controllers for 80 generations, using the same experimental set up. In each generation the goal of the robot is to navigate from an initial point to a final target point. The environment is static with obstacles in predefined positions. Each individual had 30 s to accomplish its goal and after that the experiment was terminated and the robot was repositioned. This time was enough for the robot, based on its speed, to move almost everywhere in the area which was operating from its initial point.

During the experimentation on HELOT robot we added a new parameter which takes into account the activation level of each sensor. In case that the mean activation level is less than 0.3 (experimentally derived) a penalty value was assigned to the fitness functions of the specific individual. That was done to avoid cases in which the robot has stuck to an obstacle and therefore could not complete the experiment. Obviously, all individuals who fall in this category have limited chances to survive. Notice that all the experiments were performed without dynamic obstacles since it's difficult to reproduce exactly the same conditions using real robots and not a simulation environment. The best evolved controllers were tested for various

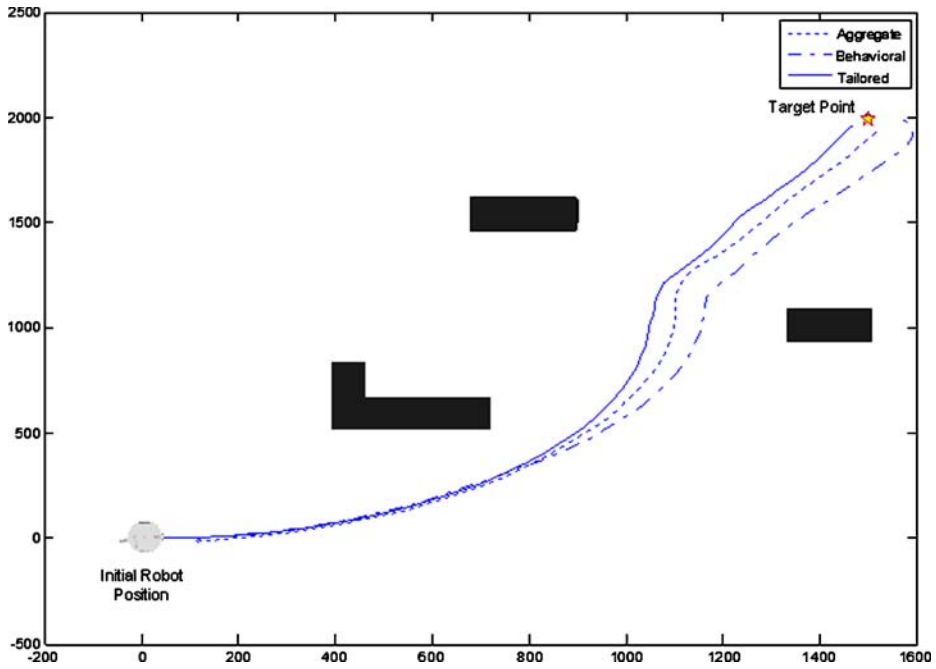

Fig. 5 Test case 1: movement from the lower left area to the upper right

Table 2 Parameters of the robot movement

Type of fitness function	Test case 1			Test case 2		
	Aggregate	Behavioral	Tailored	Aggregate	Behavioral	Tailored
Minimum distance (mm)	63.8980	65.4110	47.8566	41.8505	59.1457	42.6037
Time (sec)	27.9322	30.5126	27.7254	42.7507	30.5774	42.1305
Mean sensor activation	0.8744	0.8642	0.8799	0.8926	0.8031	0.8880
Mean lin. vel. (cm/sec)	9.39702	8.90375	9.42322	9.21670	9.00040	9.19336
Type of fitness function	Test case 3			Test case 4		
	Aggregate	Behavioral	Tailored	Aggregate	Behavioral	Tailored
Minimum distance (mm)	41.8505	59.1457	42.6037	125.3659	110.7722	122.1138
Time (sec)	42.7507	30.5774	42.1305	41.07354	40.2912	44.62946
Mean sensor activation	0.8926	0.8031	0.8880	0.816438	0.81505	0.82027
Mean lin. vel. (cm/sec)	9.21670	9.00040	9.19336	8.254323	8.152524	7.998842

navigation conditions and scenarios. Four different sets of experiments were conducted for validation, and each individual had one minute to go as close as possible to a predefined target point, without any knowledge of the area in which it was operating.

In the first test case studied, the robot was starting in the lower left of the experimental area and it was moving towards the star mark in the upper right, while avoiding different obstacles in its path. The trajectories followed by the three different controllers in test case 1, are presented in Fig. 5. The robot which used the controller derived from the tailored fitness function was the fastest and went closer to the target point (see Table 2).

In the second test case, the initial position of the robot was the upper left part while the target point was at the lower right, while avoiding different obstacles in its path. The trajectories followed by the three different controllers in test case 2, are presented in Fig. 6. The trajectory followed by the robot which is using the controller which derived from the aggregate fitness function is completely different than the other two. In the third test case, the initial position of the robot was the upper left part while the target point was at the lower right, while avoiding different obstacles in its path. The trajectories followed by the three different controllers in test case 3, are presented in Fig. 7. The robot which used the controller derived from the behavioral fitness function was slower than the other two. In the fourth test case, the initial position of the robot was the lower right part while the target point was at the upper left, while avoiding different obstacles in its path. The trajectories followed by the three different controllers in test case 4, are presented in Fig. 8.

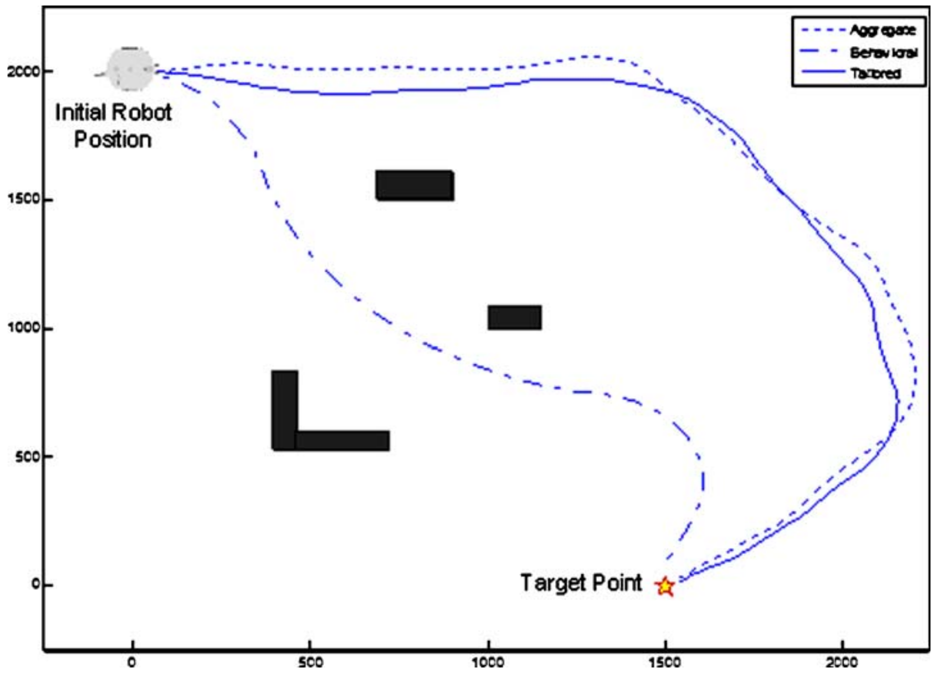

Fig. 6 Test case 2: movement from the upper left area to the lower right

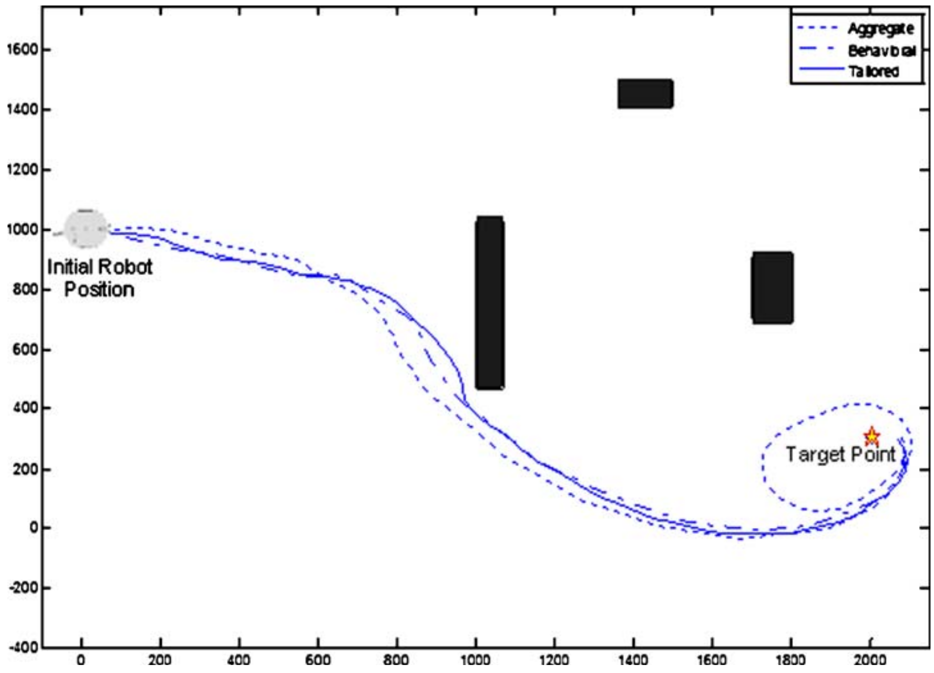

Fig. 7 Test case 3: movement from the upper left area to the lower right

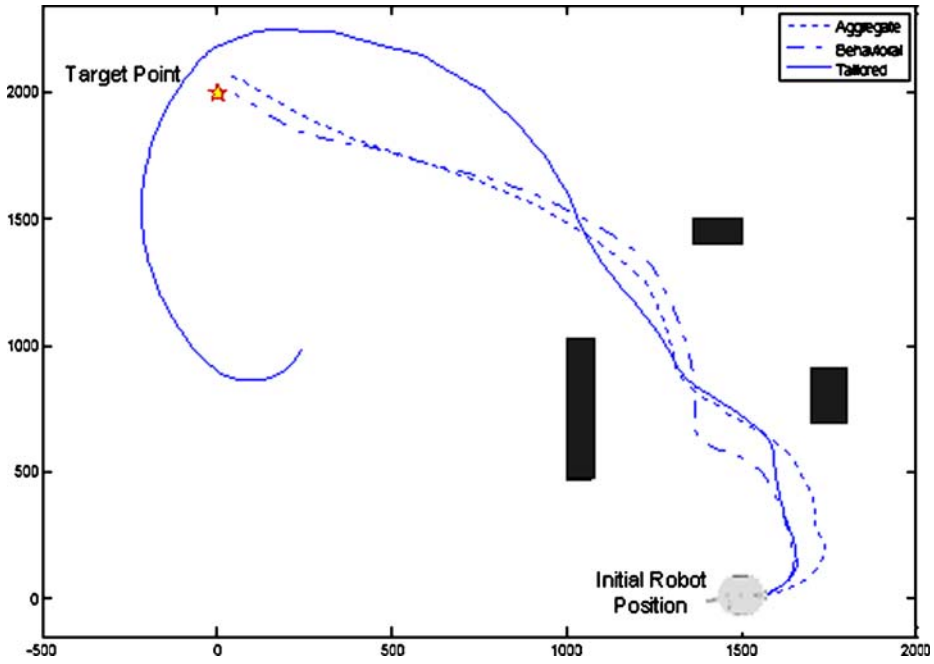

Fig. 8 Test case 4: movement from the lower left area to the upper right

Table 3 Efficiency of each fitness function for each test case

	Test case	1	2	3	4	Mean value
EF	Aggregate	3,732	3,715	3,837	3,859	3,7858
	Behavioral	3,567	3,583	3,816	3,771	3,6843
	Tailored	4	3,700	3,681	3,778	3,7898

Several parameters concerning the robot movement were monitored during their operation in the different test cases and presented in Table 2.

To measure the overall behavior of the controllers we introduce a metric of the efficiency of the robot's performance. The efficiency metric is given by:

$$\mathrm{EF}(i) = \frac{\min\{\mathrm{MD}_C\}}{\mathrm{MD}_i} + \frac{\min\{T_C\}}{T_i} + \frac{\mathrm{MSA}_i}{\max\{\mathrm{MSA}_C\}} + \frac{\mathrm{MLV}_i}{\max\{\mathrm{MLV}_C\}}, \tag{8}$$

where, MD is (final) minimum distance to target, MST is the time needed for the minimum distance, MSA is the mean sensor activation, MLV is the mean linear velocity, $c = 1,\ldots i$ is the group of robot controllers considered. The maximum value of efficiency as derives from (8) is 4. The performance of each fitness type for all test cases is presented in Table 3.

The mean efficiency in all test cases indicates that the tailored and aggregate fitness functions are likely to produce more efficient controllers in comparison to the behavioral fitness function. In most cases behavioral fitness function produces the worst results, while the aggregate and the tailored are better. The robots which are using controllers produced by the aggregate fitness functions are moving faster comparing to the others (Table 2). Although the trajectories followed are similar there are certain variations that indicate different behavior based on the fitness function used. This claim is supported by the fact that all the other parameters in the evolution process in all cases are the same. The different values of EF and the trajectories followed indicate that different fitness functions are leading to different behaviors. The fact that the efficiency of the aggregate fitness is similar to the efficiency of the tailored is indicating that efficient controllers can derive by only measuring success or failure to complete the task for which controllers are being evolved. This is promising because if aggregate selection could be achieved for much more complex tasks, it could eventually lead to the application of ER methods to environments and tasks in which humans lack sufficient knowledge to derive adequate controllers.

6 Discussion and Conclusions

Fuzzy logic controllers without appropriate tuning represent a pure reactive solution for the mobile robot navigation problem. Therefore, evolution processes have been extensively used to optimize the structural characteristics (mainly membership functions and rules) of fuzzy controllers. In these cases, the performance of the evolved controller is heavily based on the selected fitness function.

In this paper we examine the impact of the selection of the fitness function on the evolution of a fuzzy logic controller, together with the navigation performance of a real robotic vehicle. We used different types of fitness functions, namely, *aggregate, behavior and tailored*, that represent different evolution approaches. In order to evaluate the performance of the evolved controllers and investigate how fitness functions affect the overall behavior of the vehicle we introduced the efficiency factor metric. From the experiments conducted, it turned out that the "tailored" type function is more appropriate for the navigation problem in static environments.

Future research will include the formulation of a metric for the measurement of the efficiency of the performance of a team of robot vehicles. We also anticipate studying the effects of fitness function selection on controllers that avoid dynamically moving obstacles.

References

1. Tsourveloudis, N.C., Doitsidis, L., Valavanis, K.P.: Autonomous navigation of unmanned vehicles: a fuzzy logic perspective. In: Kordic, V., Lazinica, A., Merdan, M. (eds.) Cutting Edge Robotics, pp. 291–310. Pro Literatur Verlag, Mammendorf (2005)
2. Yang, X., Moallem, M., Patel, R.V.: A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation. IEEE Trans. Syst. Man Cybern., B **35**(6), 1214–1224 (2005)
3. Resu, P., Petriu, E.M., Whalen, T.M., Cornell, A., Spoelder, H.J.W.: Behavior-based neuro-fuzzy controller for mobile robot navigation. IEEE Trans. Instrum. Meas. **52**(4), 1335–1340 (2003). doi:10.1109/TIM.2003.816846
4. Tsourveloudis, N.C., Valavanis, K.P., Hebert, T.: Autonomous vehicle navigation utilizing electrostatic potentional fields and fuzzy logic. IEEE Trans. Robot. Autom. **17**(4), 490–497 (2001)
5. Ye, C., Yung, N.H.C., Wang, D.: A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. IEEE Trans. Syst. Man Cybern., B **33**(1), 17–27 (2003)
6. Lee, S.-I., Cho, S.-B.: Emergent behaviors of a fuzzy sensory-motor controller evolved by genetic algorithm. IEEE Trans. Syst. Man Cybern., B **31**, 919–929 (2001)
7. Kim, S.H., Park, C., Harashima, F.: A self-organized fuzzy controller for wheeled mobile robot using an evolutionary algorithm. IEEE Trans. Ind. Electron. **48**(2), 467–474 (2001). doi:10.1109/41.915427
8. Hagras, H., Callaghan, V., Colley, M.: Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online fuzzy-genetic system. Fuzzy Sets Syst. **141**, 107–160 (2004). doi:10.1016/S0165-0114(03)00116-7
9. Hoffman, F., Pfister, G.: Evolutionary design of a fuzzy knowledge base for a mobile robot. Int. J. Approx. Reason. **17**(4), 447–469 (1997). doi:10.1016/S0888-613X(97)00005-4
10. Matellan, V., Fernadez, C., Molina, J.M.: Genetic learning of fuzzy reactive controllers. Robot. Auton. Syst. **25**, 33–41 (1998). doi:10.1016/S0921-8890(98)00035-9
11. Nanayakkara, D.P.T., Watanabe, K., Kiguchi, K., Izumi, K.: Evolutionary learning of a fuzzy behavior based controller for a nonholonomic mobile robot in a class of dynamical environments. J. Intell. Robot. Syst. **32**, 255–277 (2001). doi:10.1023/A:1013939308620
12. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT, Cambridge (2000)
13. Nelson, A.L., Barlow, G.J., Doitsidis, L.: Fitness function in evolutionary robotics: a survey and analysis. Robot. Auton. Syst. **57**(4), 345–370 (2008)
14. Kaiser, M., Friedrich, H., Buckingham, R., Khodabandehloo, K., Tomlinson, S.: Towards a general measure of skill for learning robots. In: Proceedings of the 5th European Workshop on Learning Robots, Bari, Italy (1996)
15. Doitsidis, L., Valavanis, K.P., Tsourveloudis, N.: Fuzzy logic based autonomous skid steering vehicle navigation. In: CD-ROM Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington D.C. (2002)
16. Valavanis, K.P., Doitsidis, L., Long, M., Murphy, R.R.: Validation of a distributed field robot architecture integrated with a MATLAB based control theoretic environment: a case study of fuzzy logic based robot navigation. IEEE Robot. Autom. Mag. **13**(3), 93–107 (2006)
17. Cordon, O., Herrera, F., Hoffmann, F., Magdalena, L.: Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. World Scientific, Singapore (2001)

18. Tsourveloudis, N., Doitsidis, L., Ioannidis, S.: Work-in-process scheduling by evolutionary tuned fuzzy controllers. Int. J. Adv. Manuf. Technol. **34**(7–8), 748–761 (2007). doi:10.1007/s00170-006-0636-x
19. Michalewicz, Z.: Genetic Algorithms+Data Structures=Evolution Programs. Springer, Heidelberg (1994)
20. Floreano, D., Mondana, F.: Evolution of homing navigation in a real mobile robot. IEEE Trans. Syst. Man Cybern., B **26**(3), 396–407 (1996)
21. Nelson, A.L., Doitsidis, L., Valavanis, K.P., Long, M.T., Murphy, R.R.: Encorporation of MATLAB into a distributed behavioral robotics architecture. In: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS04), pp. 2028–2035. Sendai, Japan (2004)
22. Borenstein, J., Everett, H.R., Feng, L.: Where Am I? Sensors and Methods for Mobile Robot Positioning. Univ Michigan, Ann Arbor (1996)