Expert Systems with Applications 40 (2013) 6447–6458

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Advanced overtaking behaviors for blocking opponents in racing games using a fuzzy architecture

Luigi Cardamone [a], Pier Luca Lanzi [a,*], Daniele Loiacono [a], Enrique Onieva [b]

[a] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy
[b] Deusto Institute of Technology (DeustoTech), University of Deusto, Bilbao 48007, Spain

ARTICLE INFO

Keywords:
Fuzzy systems
Simulated car racing
Overtake
Video games
Games
Artificial intelligence
Computational intelligence

ABSTRACT

In car racing, *blocking* refers to maneuvers that can prevent, disturb or completely block an overtaking action by an incoming car. In this paper, we present an advanced overtaking behavior that is able to deal with opponents implementing blocking strategies of various difficulty level. The behavior we developed has been integrated in an existing fuzzy-based architecture for driving simulated cars and tested using The Open Car Racing Simulator (TORCS). We compared a driver implementing our overtaking strategy against four of the bots available in the TORCS distribution and *Simplix*, a state-of-the-art driver which won several competitions. The comparison was carried out against opponents implementing three blocking strategies of increasing difficulty and two different scenarios: (i) a basic scenario with one opponent on a straight stretch to overtake as quickly as possible; (ii) an advanced scenario involving a race on a non-trivial track against several opponents. The results from the basic scenario show that our strategy can *always* overtake the opponent car; in particular, our strategy is slightly more risky than the other ones and may result in a little damage, however, all the other controllers show a more careful and safe policy that often prevents them to complete an overtaking maneuver. When racing against several opponents on complex tracks, our strategy results in the best trade-off between the time spent being blocked by an opponent ahead and the number of overtaking maneuvers completed.

1. Introduction

Overtaking plays a key role in the artificial intelligence (AI) of racing games both in terms of performance, believability and players' satisfaction (Cardamone, 2012; Cardamone, Loiacono, & Lanzi, 2009c). Alas, overtaking behaviors are very difficult to code (even in high-end games Lecchi, 2009) as they have to manage a large number of variables (e.g., the current trajectory, position, speed and acceleration of the driver's car and the nearby opponents, incoming bends, etc.,) and a variety of unexpected, challenging, and dangerous situations (e.g., change in opponents' trajectory, position, and blocking maneuvers, etc.).

Blocking maneuvers are strategies that a driver can use to prevent, disturb or possibly block an overtaking action by an incoming car and they are frequently employed in racing games both by human players and non-playing characters. They are typically performed by adapting the vehicle trajectory so as to block the racing line chosen by an incoming vehicle. Blocking maneuvers are typically simple to implement and can block even the most advanced overtaking strategies. A preliminary analysis we performed (Onieva, Cardamone, Loiacono, & Lanzi, 2010) on seven of the most competitive drivers available for TORCS shows that while most of these competitive drivers implement reliable overtaking strategies, their performance is dramatically reduced when facing opponents implementing even simple blocking strategies. Even the most advanced human-coded drivers (e.g., the winner of the 2009 TORCS Endurance World Championship) fails in dealing with basic blocking strategies.

In this paper, we present an advanced overtaking behavior, implemented using a fuzzy system, that can overtake challenging opponents using blocking strategies of increasing difficulty. We evaluated our overtaking behavior using two different scenarios: (i) a basic scenario with one opponent on a straight stretch to overtake as quickly as possible; (ii) an advanced scenario involving a race on a non-trivial track against several opponents. We considered opponents implementing three blocking strategies: (i) *limited blocking*, which adapts to the opponent's trajectory but avoids going too near to the track borders (so that the incoming car has still some chance of completing the overtaking maneuver); (ii) *slowly reactive blocking*, which adapts to the opponent's trajectory with a delay of one second but has no limitation and can completely block the overtaking maneuver; (iii) *fully reactive blocking*, which adapts to the opponent's trajectory with no delay and no limitation. This work extends our preliminary study (Onieva

* Corresponding author. Tel.: +39 0223993472; fax: +39 0223993411.
E-mail addresses: cardamoneluigi@gmail.com (L. Cardamone), pierluca.lanzi@polimi.it (P.L. Lanzi), daniele.loiacono@polimi.it (D. Loiacono), enrique.onieva@deusto.es (E. Onieva).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.04.030

et al., 2010) where we presented a much simpler driver that could only drive and overtake one opponent on a straight stretch. We compared our driver using our advanced overtaking behavior against five of the best drivers available for The Open Car Racing Simulator TORCS. Our results show that in the simpler scenarios our strategy can *always* overtake the opponent car; in particular, our strategy appears to be slightly more risky and may cause a little damage to the car while achieving 100% success; in contrast, more careful handcoded controllers demonstrate a safer policy that cause no damage to the car but too often do not complete any maneuver. Interestingly, our driver demonstrates an interesting emerging behavior in which it deceives the blocking opponent by initially moving to one side of the track (and thus making the opponent move on the same side) before overtaking on the opposite side. Our results in the more complex racing scenarios show that our strategy provides the best trade-off between the time spent being blocked by an opponent ahead and the number of overtaking maneuvers completed.

2. Related work

Car racing games are a popular testbed for methods of computational intelligence (Butz & Lonneker, 2009; Cardamone, Loiacono, & Lanzi, 2009a, 2009b, 2009c; Ebner & Tiede, 2009; van Hoorn, Togelius, Wierstra, & Schmidhuber, 2009; Munoz, Gutierrez, & Sanchis, 2009; Onieva, Pelta, Alonso, Milanes, & Perez, 2009; Perez, Recio, & Saez, 2009; Fujii, Nakashima, & Ishibuchi, 2008; Ho & Garibaldi, 2008a; Loiacono et al., 2008). This is probably due both to the availability of several open-source frameworks with realistic physics and engaging graphics (TORCS; Venzon; Wolf-Dieter et al.) and also to the many (10–20) competitions on simulated car racing orgazined since 2007 at major conferences (e.g., IEEE CEC 2008–2010, ACM GECCO 2009–2012, IEEE CIG 2009–2012, EvoStar 2011–2012, Loiacono et al., 2008, 2010b). Most of the works published in this area focus either (i) on the development of complete drivers using a wide variety of methods (e.g., neural networks Cardamone et al., 2009a, 2009c, fuzzy logic Onieva et al., 2009; Perez et al., 2009; Fujii et al., 2008; Ho & Garibaldi, 2008a, 2008b, evolutionary algorithms Ebner & Tiede, 2009, supervised learning Cardamone, Loiacono, & Lanzi, 2009b; van Hoorn et al., 2009; Munoz et al., 2009; Quadflieg, Preuss, Kramer, & Rudolph, 2010); or (ii) on the parameter optimization of human-designed bots (Butz & Lonneker, 2009; Butz, Linhardt, & Lonneker, 2011; Quadflieg, Preuss, & Rudolph, 2011; Preuss, Quadflieg, & Rudolph, 2011) and car setup (Wloch & Bentley, 2004; Cardamone, Loiacono, & Lanzi, 2010; Kemmerling & Preuss, 2010). In this section, we provide a brief overview of the published works related to this study.

2.1. Fuzzy systems for car racing

Fuzzy systems have been seldom used in car racing games. Onieva et al. (2009) developed a modular architecture in which the general driving was implemented by a fuzzy system controlling the target speed; the overtaking behavior was implemented by a separate heuristics that modified the target speed when opponents were detected. In Ho and Garibaldi (2008b), introduce the concept of Context-Dependent fuzzy system, in which the membership functions of the fuzzy variables are not fixed but change according to the context. The proposed approach is applied to design a controller for the car racing competition held at FuzzIEEE 2007. In Ho and Garibaldi (2008a), the same authors, present an improved version of the controller for the 2007 CIG Simulated Car Racing Competition. The driver has a two-layers architecture that combines a high-level path planner with a low-level execution controller based on fuzzy logic. In Fujii et al. (2008), fuzzy rules are generated from a set of training patterns; the study compares two methods for generating such training patterns and two representations of the sensory information (third person vs. egocentric). In Perez et al. (2009), present a driver based on a fuzzy controller for the 2008 CIG Simulated Car Racing Competition. First, they designed the rules and the fuzzy sets of a base driver. Then, they applied a genetic algorithm to optimize the parameters of the fuzzy sets. In Onieva et al. (2010), we presented an initial study of blocking in car racing games based on our experience in the organization of the Simulated Car Racing Competition (Loiacono et al., 2008, 2010b); we showed that even the most competitive controller can fail to overtake even the most basic blocking strategies on very simple straight track sections; we also showed that a simple fuzzy controller could tackle blocking behaviors that more advance drivers failed to manage.

2.2. Overtaking behaviors in car racing games

Although overtaking strategies are known to play a key role in the development of competitive drivers (Cardamone et al., 2009c; Butz & Lonneker, 2009; Butz et al., 2011), few published works focus on this topic.

Cardamone et al. (2009c) applied neuroevolution (more precisely NEAT Stanley, 2004) to evolve a competitive driver capable of overtaking in complex situations. The architecture comprised one neural network for the driving alone and one neural network for overtaking. The two networks were evolved separately on different scenarios. The network for overtaking was activated on top of the main driving behavior (in a sort of behavior-based architecture) when an opponent close to the car was detected. Loiacono, Prete, Lanzi, and Cardamone (2010a) applied simple reinforcement learning to learn separate strategies for (i) overtaking an opponent on a straight stretch by exploiting the drag effect; and for (ii) overtaking an opponent close to a turn using braking delay. Butz et al. (2011) developed a driver for TORCS based on a sensory-to-motor policy, optimized using an evolutionary strategy with Covariance Matrix Adaptation (CMA-ES). Overtaking was implemented as a module that (i) monitored the opponents position and speed around the driver and (ii) projected the opponents position onto the driver's track sensors. Accordingly, the opponents were perceived as obstacles on the track and the overtaking actions were performed by the basic driving module.

3. TORCS

The Open Racing Car Simulator (TORCS) is a state-of-the-art open source car racing simulator which provides a sophisticated physics engine, full 3D visualization (see Fig. 1), several tracks and models of cars, and different game modes (e.g., practice, quick race, championship, etc.). The car dynamics is simulated by a powerful physics engine that takes into account many aspects of racing cars such as traction, aerodynamics, fuel consumption, etc. Each car is controlled by an automated driver or *bot*. At each control step, a bot can access the current game state, which includes several information about the car and the track as well the information about the other cars on the track, and can control the car using the gas/brake pedals, the gear stick, and steering wheel. The game distribution includes several programmed bots which can be easily customized or extended to build new bots.

4. Overtaking and blocking behavior

Driving a racing car on a track alone is a task easy to program and to learn using computational intelligence methods. In contrast, driving a racing car against other opponents is very complex be-

Fig. 1. A screenshot from TORCS.

cause of the unpredictable actions of the opponents and the possible interactions between the different drivers' strategies. Accordingly, overtaking is one of the most challenging task when developing the artificial intelligence of a racing game (Lecchi, 2009). In fact, overtaking has to take into account the opponents' driving strategies and also elude possible *blocking* maneuvers performed by the opponents to prevent the overtake.

In this work, we propose a framework to design overtaking behaviors capable of dealing with complex scenarios involving opponents that implement three different blocking strategies of increasing difficulty: *slowly reactive*, *limited*, and *fully reactive*. Drivers implementing a *slowly reactive* blocking strategy change their direction only once every second on the basis of the current position of the overtaker on the track. This strategy leads to a rather realistic behavior that, instead of mimicking the opponent trajectory, reacts to the overtaker actions with sudden changes of direction after a reasonable delay. Drivers implementing a *limited* blocking strategy mimic the trajectory followed by the incoming driver (that tries to overtake) but, to avoid possible collisions, keep a safety distance from the borders of the track that is larger than the width of a car. Therefore, despite reacting very quickly to the overtaker actions, opponents with limited blocking capabilities always leave the overtaker the chance to pass. Finally, drivers implementing a *fully reactive* blocking strategy mimic the trajectory of the overtaker exactly but their actions are not limited nor delayed. Accordingly, they can both block the incoming car completely and may also cause collisions.

5. Our framework

We extended the framework introduced by Onieva et al. (2009) to develop a controller capable of dealing with complex overtaking scenarios. More specifically, we designed a modular controller with a three-layers structure (Fig. 2): a perception layer that acquires data from the sensors (e.g., the status of the driver's car, the perceived position, speed and trajectory of the opponents, the track, etc.); a decision layer that processes the data provided by the perception layer and selects an action to perform (e.g., brake, accelerate, overtaking, recovering from a crash, etc.); an actuation layer that performs the selected action using the available effectors (e.g., the throttle, the brake, the gearbox, the steering wheel, etc.).

5.1. Perception layer

This layer collects data from the environment surrounding the car and pre-processes them to provide suitable information to the decision layer. The underlying sensory model is the same one used for the Simulated Car Racing Championships (Loiacono, Cardamone, & Lanzi, 2009, 2010b) which includes an array of range-finders that perceives the nearby track limits as well as the nearby opponents; the current speed, the engine's RPM, the current gear, the orientation of the car on the track, the fuel level, etc.[1] In particular, the perception layer use the sensory inputs to compute four variables that are sent to the decision layer: (i) the forward distance (d) from the closest opponents along the track (see Fig. 3); (ii) the *expected collision time* (τ) computed as the estimated time to reach the closest opponent assuming that both cars maintain the current speed; (iii) the *lateral distance* (δ) from the closest opponent with respect to the section of the track which is positive when the opponent is on the driver's left side and negative when it is on the driver's right side; (iv) the position (p) of the car on the track computed as the relative distance from the track axis (p is 0 when car is in the middle of the track, -1 when the car is on the left edge, and $+1$ when it is on the right edge).

5.2. Decision layer

This layer selects the next control actions based on the inputs provided by the perception layer. It consists of several modules that manage different tasks (Fig. 2) such as plain driving, recovery from incidents, overtaking, etc. In this work, we focused on the overtaking task and implemented the corresponding module as a fuzzy system based on the computational model of fuzzy co-processor ORBEX (Garca & de Pedro, 1998). Accordingly, the overtaking strategy is designed as a set of *if-then* rules expressed in a quasi-natural language, while the t-norm *minimum* and the t-conorm *maximum* are used to implement *and* and *or* operators. As in Onieva et al. (2009), our fuzzy system (i) represents inputs using trapezoidal membership functions and (ii) singletons (Takagi & Sugeno, 1985) to represent the outputs; (iii) it applies Mamdani-type inference (Mamdani, 1974); and (iv) uses the center of gravity/mass for defuzzification.

Input variables. The four continuous input variables provided from the perception layer are mapped into four linguistic variables through trapezoidal membership functions. Fig. 4 shows the membership functions used to encode the distance from the closest opponent d: (i) `back` encodes the end of the overtake, when the opponent is behind the controlled car and the maneuver can be finalized; (ii) `parallel` encodes the most critical stage of the overtake, when the controlled car and an opponent are side by side; (iii) `close` encodes the beginning of the overtake, when the controlled car is approaching an opponent. Fig. 5 shows the membership functions used to encode the expected collision time τ: (i) `normal` corresponds to a completely safe situation, when no collision can happen in the near future; (ii) `warning` corresponds to a safe situation that might lead to a dangerous situation in the near future; (iii) `danger` corresponds to an eventually dangerous situation when a collision is imminent; (iv) `negative` corresponds to a collision (encoded by a negative expected collision time).

Fig. 6 shows the membership functions to encode the lateral distance δ as five values: $\texttt{atRight}_2$, $\texttt{atRight}_1$, `centered`, $\texttt{atLeft}_1$, and $\texttt{atLeft}_2$. Finally, Fig. 7 shows the membership function that encodes the position of the car on the track p in eight regions, from the leftmost $\texttt{left}_4$ to the rightmost $\texttt{right}_4$.

Output variables. The decision layer computes two output variables $\hat{p}$ and $\hat{a}$ to regulate the overtaking behavior of the controlled car: $\hat{p}$ represents the target position of the car and it is encoded using

[1] Note that this sensor model is more realistic than the one available in the original TORCS distribution, which provides *exact* information about the status of the entire environment including the opponents' cars.

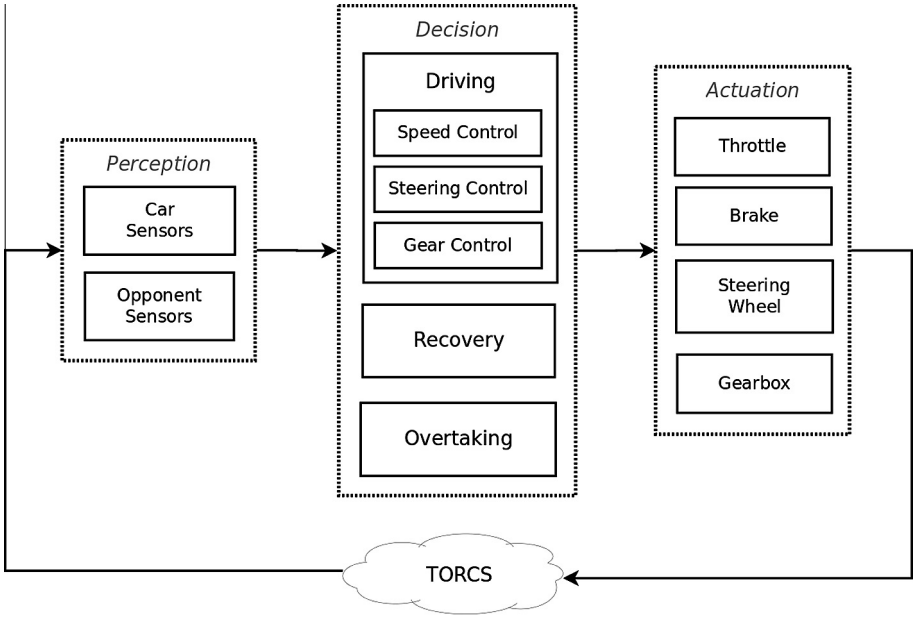


Fig. 2. Overview of the controller's architecture.

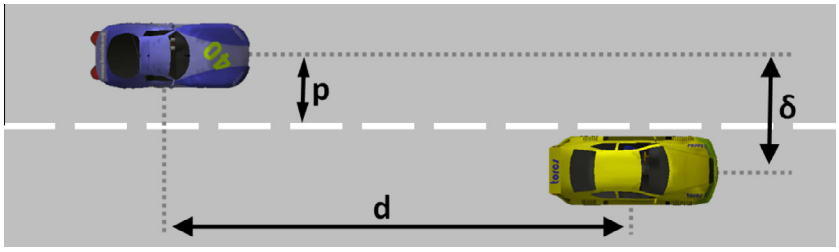


Fig. 3. The forward distance d, the lateral distance δ and the position p.

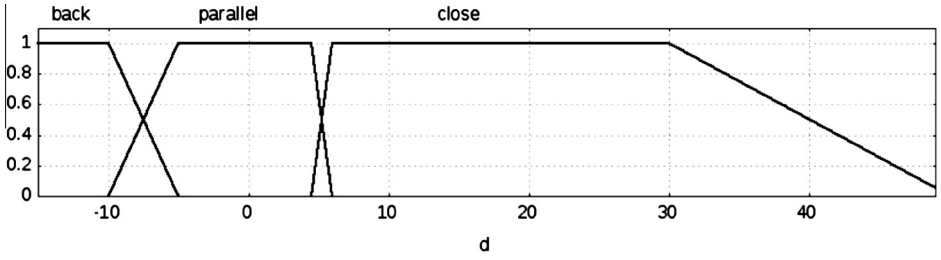


Fig. 4. Membership functions used to encode d.

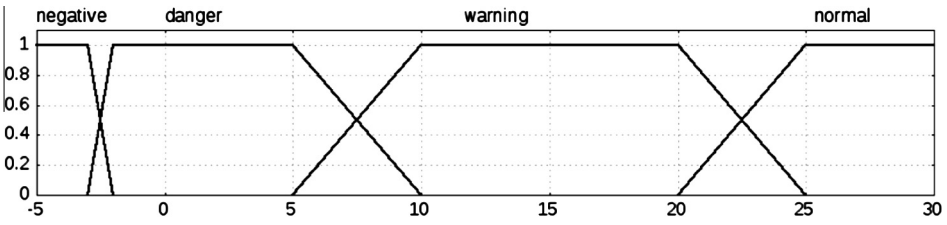


Fig. 5. Membership functions used to encode τ.

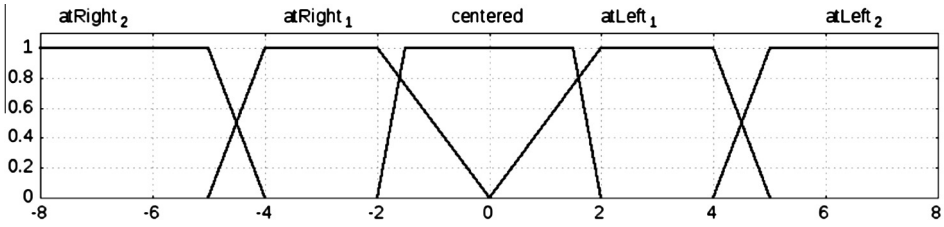


Fig. 6. Membership functions used to encode δ.

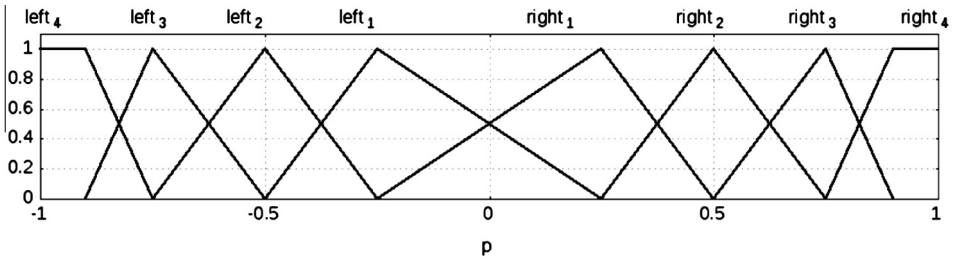


Fig. 7. Membership functions used to encode p.

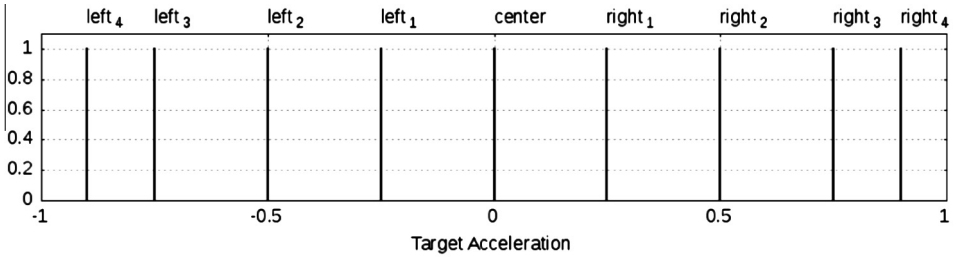


Fig. 8. Singletons used to encode $\hat{p}$.

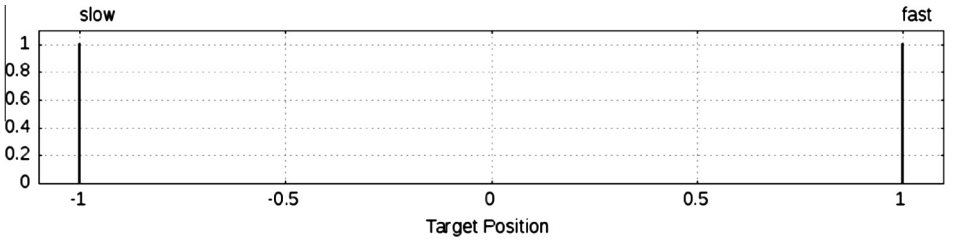


Fig. 9. Singletons used to encode $\hat{a}$.

nine symmetrical singletons (Fig. 8) each one corresponding to a target position; $\hat{a}$ represents the target acceleration of the car and it is encoded with two singletons (Fig. 9) corresponding to full acceleration (`fast`) and to a full deceleration (`slow`). These output variables are defuzzified before being sent to the actuation layer.

Rule base. The overtaking behavior is represented as a set of *if-then* rules, each one representing a small piece of the overall solution: the rule premise (the *if* clause) identifies when the rule should be applied; the rule consequence (the *then* clause) advocates one or more control decisions. As an example, Table 1 provide a tabular representation of the rule base that computes the target position $\hat{p}$ to start an overtaking maneuver when the time to collision τ corresponds to the fuzzy value `warning`: given the current car position on the track (column p) and the current lateral distance (upper row δ), the corresponding table position contains the fuzzy value of the target position $\hat{p}$. Thus, each table position corresponds to a fuzzy rule in the rule base. The table reports the target position of the car computed for each configuration of the input variables δ and p. The rules move the car toward to the track edge that results in an increase in *lateral* distance from the opponent. Note that the behavior in Table 1 is not completely symmetrical to avoid any cyclic behavior that might prevent the overtake.

Table 2 shows the rules activated to avoid an imminent collision in the most critical stage of the overtake, when the forward distance d is `close` and the time to collision τ is `danger`. Table 2 reports the target position $\hat{p}$ and the target acceleration $\hat{a}$ computed

Table 1
Rules applied when τ is `warning`. The table reports the target position $\hat{p}$ for each possible value of the input variables δ and p.

p	δ				
	$\texttt{atRight}_2$	$\texttt{atRight}_1$	centered	$\texttt{atLeft}_1$	$\texttt{atLeft}_2$
$\texttt{left}_4$	$\texttt{left}_4$	$\texttt{left}_3$	$\texttt{left}_3$	center	$\texttt{left}_4$
$\texttt{left}_3$	$\texttt{left}_3$	$\texttt{left}_2$	$\texttt{left}_2$	center	$\texttt{left}_4$
$\texttt{left}_2$	$\texttt{left}_2$	$\texttt{left}_2$	$\texttt{right}_2$	center	$\texttt{left}_3$
$\texttt{left}_1$	$\texttt{left}_1$	center	$\texttt{right}_2$	$\texttt{left}_2$	$\texttt{left}_2$
$\texttt{right}_1$	$\texttt{right}_2$	$\texttt{right}_2$	$\texttt{right}_2$	center	$\texttt{right}_1$
$\texttt{right}_2$	$\texttt{right}_3$	center	$\texttt{right}_2$	$\texttt{right}_1$	$\texttt{right}_2$
$\texttt{right}_3$	$\texttt{right}_4$	center	$\texttt{right}_2$	$\texttt{right}_2$	$\texttt{right}_3$
$\texttt{right}_4$	$\texttt{right}_4$	center	$\texttt{right}_3$	$\texttt{right}_3$	$\texttt{right}_4$

Table 2
Rules applied when τ is `danger` and d is `close`. The table reports the target position $\hat{p}$ and the target acceleration $\hat{a}$ for different configurations of the input variables δ and p.

p	δ		
	$\texttt{atRight}_1$	centered	$\texttt{atLeft}_1$
$\texttt{left}_4$	center + slow	center + slow	$\texttt{right}_1$ + fast
$\texttt{left}_3$	center + slow	center + slow	$\texttt{right}_1$ + fast
$\texttt{left}_2$	$\texttt{left}_4$ + fast	$\texttt{right}_4$ + slow	$\texttt{right}_1$ + fast
$\texttt{left}_1$	$\texttt{left}_4$ + fast	$\texttt{right}_4$	$\texttt{right}_1$ + fast
$\texttt{right}_1$	$\texttt{left}_1$ + fast	$\texttt{right}_4$	$\texttt{right}_4$ + fast
$\texttt{right}_2$	$\texttt{left}_1$ + fast	$\texttt{left}_2$ + slow	$\texttt{right}_4$ + fast
$\texttt{right}_3$	$\texttt{left}_1$ + fast	center + slow	center + slow
$\texttt{right}_4$	$\texttt{left}_1$ + fast	center + slow	center + slow

by the rule set based on the current car position on the track (column p) and the current lateral distance (upper row δ). Thus, each table position corresponds to a fuzzy rule.

Table 2 shows that overtake is aborted (situation represented by a `slow` target acceleration) when the car is very close to a track edge. Overtake is also aborted when the lateral distance δ from the opponent is very small (i.e., τ is `center`) and also when the opponent's position is blocking the ideal overtaking line (e.g., when p is $\texttt{left}_4$ and δ is $\texttt{left}_1$). Conversely, when the ideal overtaking trajectory is available the overtake is completed as soon as possible (situation represented by a `fast` target acceleration in Table 2). For the sake of brevity, Table 2 does not include the case when δ is $\texttt{atLeft}_2$ nor the case δ is $\texttt{atRight}_2$ that are not critical for the successful completion of the overtake maneuver (in fact, the lateral distance is rather large).

These examples show that our framework is flexible and allows to design complex behaviors easily without requiring a detailed background knowledge of the environment and its underlying representation.

5.3. Actuation layer

This layer maps the actions selected by the decision layer into control actions that are performed through the effectors. When the overtaking subsystem outputs the suggested actions, the decision layer (i) maps the output variables $\hat{p}$ and $\hat{a}$ (Fig. 2) into control actions and (ii) merges them with the actions advocated by the *driving subsystem* (Fig. 2). In particular, the target position $\hat{p}$ is mapped into a steering command computed as,

$$steer = \frac{\alpha + 0.5 \cdot (\hat{p} - p)}{\beta},$$

where α is the angle between the car direction and the track axis, β is the maximum turn angle of the car. The mapping of the target acceleration $\hat{a}$ into a control action is straightforward: a positive value (i.e., when $\hat{a}$ is `fast`) results in an acceleration action, while a negative value (i.e., when $\hat{a}$ is `slow`) results in a braking action. To merge the control actions from the overtaking and driving subsystems (i) the steering action advocated by the overtaking subsystem is added the steering action computed by the driving subsystem; (ii) the acceleration action advocated by the overtaking subsystem is reduced by the 25% before being added to the acceleration action computed by the driving subsystem; similarly, (iii) the braking action is reduced by 75% before being added to corresponding action from the driving subsystem.

6. Basic overtaking scenario

In the first set of experiments, we compared the overtaking capabilities of our driver against the best five TORCS drivers publicly available. We considered a basic overtaking scenario involving a long straight stretch and an opponent that has to be overtaken in the shortest time possible.

6.1. Design of experiments

Task description. All the experiments were performed on a straight stretch, 15 m wide and 2800 m long, with (i) one car controlled by an overtaking driver to evaluate and (ii) one car controlled by the opponent; both drivers used the same car model, *car1-trb1*, available in the TORCS distribution (TORCS). At the beginning of an experiment, both cars were positioned in the center of the track; the overtaking car (controlled by the driver to be evaluated) was d meters behind the opponent; both cars had an initial speed of 170 km/h. The experiment ended when either (i) the opponent reached the end of the straight stretch without being overtaken (that is, when the overtaking maneuver failed); or (ii) the overtaking car was successfully completed the maneuver and the overtaking car was 50 m ahead of the opponent. We evaluated each driver using 12 different values of distance d ($d \in \{80, 100, 120, \ldots, 300\}$). The minimum value of d was set to guarantee that even the slowest driver considered could reach a speed sufficient to complete the overtaking maneuver by reaching a speed that was at least 50 km/h faster than the opponent.

Evaluation metrics. For each experiment, and for each driver, we measured the percentage of overtakings successfully completed; the time required to complete successful maneuvers; and the percentage of overtakings involving collisions.

Opponents strategies. The opponent driver was programmed to maintain the initial speed equal to 170 km/h and to drive in the center of the track. To make the overtaking task realistic, the opponent also implemented a blocking behavior to prevent an overtake manuever by repeatedly changing direction so as to block the trajectory of the incoming car. We implemented three different blocking strategies of increasing difficulty: *slowly reactive*, *limited*, and *fully reactive*. The opponent activated the blocking behavior as soon as the incoming vehicle was less than 75 m away.

An opponent implementing a *limited* blocking strategy mimics the trajectory followed by the overtaker but, to avoid possible collisions, it keep a safety distance from the borders of the track (which is larger than the width of a car). Therefore, despite reacting very quickly to the overtaker actions, an opponent with limited blocking capabilities always leaves the overtaker a chance to pass. An opponent implementing a *slowly reactive* blocking strategy changes its direction only once every second on the basis of the current position of the overtaker on the track. This strategy leads to a rather realistic behavior that, instead of mimicking the opponent trajectory, reacts to the overtaker actions with a reasonable delay. Finally, an opponent implementing a *fully reactive* blocking strategy mimics the trajectory of the overtaker exactly but its ac-

Table 3
Overtaking an opponent with the *limited* behavior.

Driver	Berniw	Bt	Fuzzy	Inferno	Olethros	Simplix
Success	0.0%	0.0%	100.0%	0.0%	0.0%	100.0%
Collision	0.0%	0.0%	8.3%	0.0%	0.0%	0.0%
OvertakingTime	–	–	14.8	–	–	12.9

Table 4
Overtaking an opponent with the *slowly reactive* behavior.

Driver	Berniw	Bt	Fuzzy	Inferno	Olethros	Simplix
Success	91.7%	58.3%	100.0%	100.0%	83.3%	0.0%
Collision	100.0%	41.7%	16.7%	100.0%	16.7%	0.0%
OvertakingTime	33.4	18.5	14.0	32.2	14.3	–

tions are not limited nor delayed. Accordingly, it can block the incoming car completely but may also cause a collision.

6.2. Experimental results

We compared the overtaking capabilities of our controller against the best four drivers publicly available in the TORCS distribution (*Berniw*, *Bt*, *Inferno*, *Olethros*) and *Simplix*, a state-of-the-art driver winner of the 2009 TORCS Endurance World Championship (Cardamone et al., 2010). All the drivers were evaluated against three types of opponents implementing a *limited*, a *slowly reactive*, and a *fully reactive* blocking strategy.

Limited blocking. Table 3 reports the performance of our controller (reported as *Fuzzy*) and the other five drivers considered in this work against an opponent with limited blocking. For each driver, Table 3 reports the overtaking time as an average over all the successfully completed overtakes. Although limited blocking implements a rather basic strategy, all the drivers available in the TORCS distribution fail to complete any overtake because of their safe driving policies. The analysis of their trajectories show that these controllers tend to remain behind the opponent and follow the blocking vehicle, without causing any collision with it. Only our controller (*Fuzzy*) and *Simplix* (the best driver publicly available), can overtake an opponent with *limited* blocking. In particular, both *Fuzzy* and *Simplix* achieve a 100% success rate. *Simplix* can overtake in all the experiments without causing any collision while our controller had few collisions (that still allowed it to complete the overtake) only in one out of the 12 experiments performed, namely when the initial distance d is 120 m. In terms of overtaking time, both drivers are able to complete the maneuver in a rather short time span; *Simplix* is slightly faster than our controller. This happens because the more sophisticated policy that our driver implements requires more direction changes than *Simplix*.

Fig. 10 compares the trajectories of our controller (upper) and *Simplix* (lower) in the same overtaking setup; the labels report the time-stamp (in seconds) and the red lines represent the occluding vehicle. As can be noted, *Simplix* can successfully overtake the opponent because it immediately moves to one side of the track and maintains the same trajectory until the overtake is completed. In contrast, our driver *Fuzzy* performs a more sophisticated maneuver to get around the blocking: at the beginning, until time-stamp 8, *Fuzzy* performs the typical overtaking maneuver and when the opponent is on the same trajectory our driver changes direction and starts the overtaking; as *Fuzzy* is getting closer, the opponent starts its blocking action (time-stamp 10); at this point, our controller moves in the opposite direction and when the opponent performs another blocking action, our controller changes direction again (at time-stamp 12) following a wider trajectory that brings it slightly offtrack but results in the completion of the overtaking action. Overall, our driver, *Fuzzy*, implements a rather advanced and realistic strategy with a deceiving maneuver that basically entangles the opponent during the second wider direction change.

Slowly reactive blocking. Table 4 reports the performance of all the drivers considered in this study against the opponent implementing a slowly reactive blocking. As can be noted, all the drivers except *Simplix* can exploit the delayed reaction of the opponent and can thus complete some overtakes. However, the same delay is also the cause of the many collisions for all of the drivers that can complete some overtakes: only *Simplix* can avoid any collision but at the same time it cannot complete any overtake because of its very cautious driving policy. In terms of overtaking time, *Fuzzy* demonstrates the best performance requiring on the average only 14.0 s to complete an overtaking maneuver. *Olethros* shows a very competitive overtaking time of 14.3 s. *Berniw* and *Inferno* are the slowest controllers, requiring more than 30 s (more than twice the time required by *Fuzzy*). Overall, the results suggest that our controller, *Fuzzy*, may provide the best trade-off as it can complete all the overtaking maneuvers quickly while being involved in few collisions.

Fig. 11 compares the overtaking behaviors of *Fuzzy*, *Olethros*, and *Berniw* against the slowly adaptive opponent when the initial distance (d) is 180 m. As can be noted, our controller (top) completes the overtaking in the lesser time while keeping a rather safe

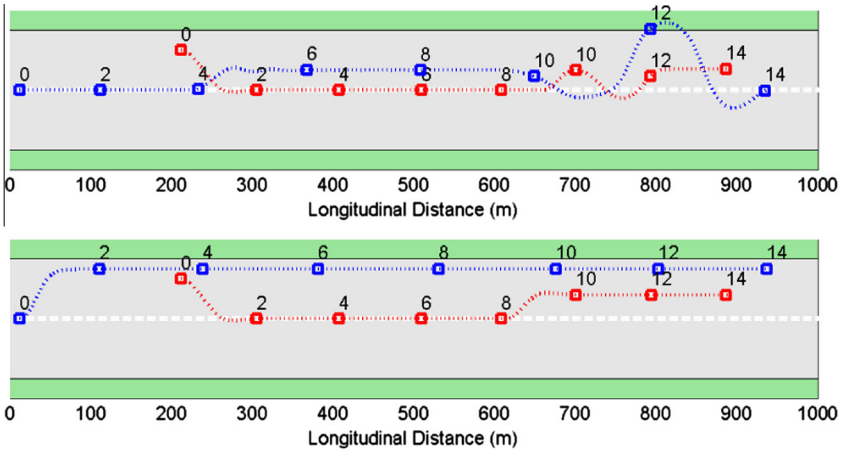


Fig. 10. Trajectories of our controller, *Fuzzy*, (top) and *Simplix* (bottom) against the *limited* opponent and initial distance d of 200 m.

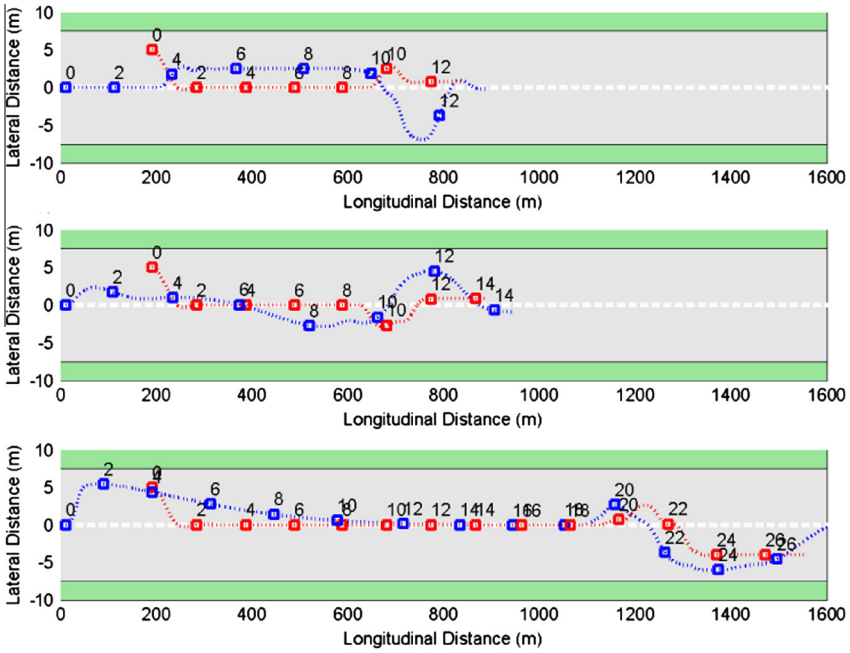


Fig. 11. Trajectories of the *Fuzzy* (top), *Olethros* (center) and *Berniw* (bottom) against the *slowly reactive* opponent and initial separation of 180 m.

Table 5
Overtaking an opponent with the *fully reactive* behavior.

Driver	Berniw	Bt	Fuzzy	Inferno	Olethros	Simplix
Success	0.0 %	0.0 %	100.0 %	0.0 %	0.0 %	0.0 %
Collision	0.0 %	0.0 %	16.7 %	0.0 %	0.0 %	0.0 %
OvertakingTime	–	–	16.1	–	–	–

lateral distance during the maneuver; *Olethros* is only a little bit slower but very competitive with respect to our driver; *berniw* is much slower and drives too close to the opponent so that it caused a collision during the maneuver.

Fully reactive blocking. Table 5 reports the performance of all the drivers considered against the opponent implementing the most challenging fully blocking strategy fully. As can be noted, our controller (*Fuzzy*) is the only one capable of completing all the overtaking maneuvers, receiving limited damage in one case, while all the other controllers (even the very advanced *Simplix*) cannot complete an overtaking not even once.

Fig. 12 compares the trajectory followed by our controller (top), *Olethros* (second from top), *Berniw* (third from top) and *Simplix* (bottom). As can be noted, our controller has a highly realistic behavior when compared to the bots distributed with TORCS (*Olethros* and *Berniw*) and the more competitive *Simplix*. As previously shown, our controller initially tries a rather standard overtaking maneuver (until time-stamp 6), as the blocking behavior begins, our driver tries to deceive the opponent by initially moving to the right and then to the left with a wider trajectory which allows for the completion of the maneuver. In contrast, all the other controllers cannot cope with the quick changes in the opponent strategy and they all end up following the car ahead without completing any overtaking maneuver.

6.3. Discussion

Table 6 summarizes the results of the experiments involving the three blocking strategies; for each driver, the table reports the average performance achieved by each controller against the three blocking strategies. As can be noted, only our controller (*Fuzzy*) can overtake all the three opponents considered reaching 100% performance while all the other drivers can overtake only in less than 40% of the cases. Such low performance is either due to a cautious driving policy (as in the case of *Simplix* and *Olethros* which causes almost no collision) or to the inability to deceive the blocking strategy (as in the case of the fully blocking strategy when no other driver can overtake). Overall, our fuzzy controller takes some risks and receives some damage but at the same time it allows the completion of all the maneuvers so as to provide the best trade-off between the overtaking capabilities and the damage suffered.

7. Advanced overtaking scenario

In the previous set of experiments, we evaluated the overtaking capabilities of our driver in a basic scenario that did not include most of the sources of bias present in actual races (e.g., the track shape, the interactions among different driving policies). In the second set of experiments, we considered a more realistic scenario involving a race against more opponents on non-trivial tracks. This setup introduces several new challenges in that (i) overtakes can happen anywhere on the track (on a bend, on a straight stretch, or inbetween them) and (ii) they typically involve more opponents and thus interactions among more drivers.

7.1. Design of experiments

Task description. Each experiment consisted of a ten laps race against ten opponents. Initially, the opponents are uniformly distributed along the track to simulate a mid-race scenario. To limit the sources of bias, all the drivers had the same car model used in the previous set of experiments (*car1-trb1* in the TORCS distribution) and all the opponents implemented the same driving and blocking policy (*limited*, *slowly reactive* or *fully reactive*). Since all the opponents behave the same, they tended to remain uniformly distributed along the track until they started interacting with the controller under evaluation during overtaking maneuvers. Accord-

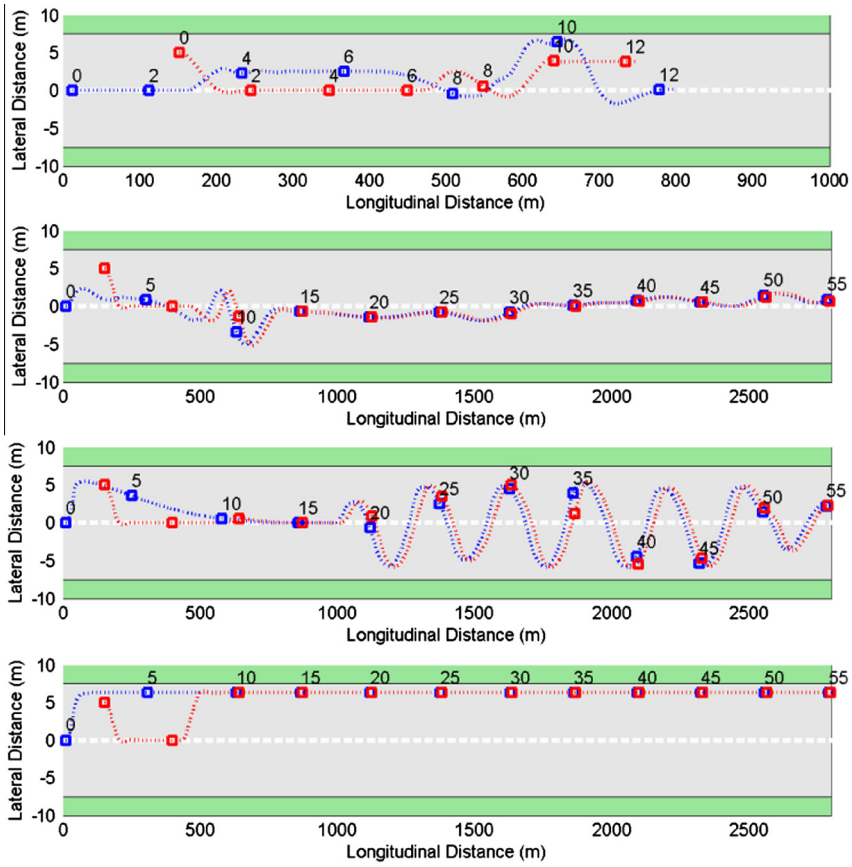


Fig. 12. Trajectories of *Fuzzy* (top) *Olethros* (second from top), *Berniw* (third) and *Simplix* (bottom) against the *fully reactive* opponent with an initial distance *d* of 140 m.

Table 6
Overall overtaking performance.

Driver	Berniw%	Bt%	Fuzzy%	Inferno%	Olethros%	Simplix%
Success	30.6	19.4	100.0	33.3	27.8	33.3
Collision	33.3	13.9	13.9	33.3	5.6	0.0

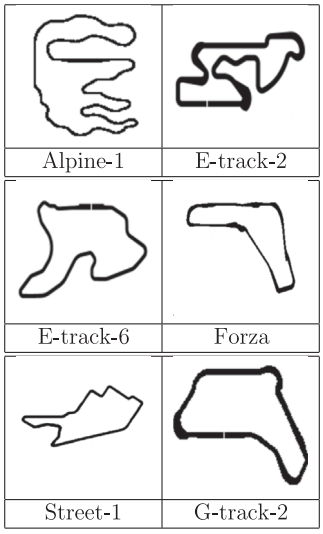
ingly, the evaluated driver will initially overtake one opponent at the time in different sections of track. Then, as the initial opponents' distribution is gradually lost (because of the interactions with the evaluated controller), scenarios become more complex with more blocking opponents to overtake in rapid succession or even at the same time.

Evaluation metrics. The evaluation of an overtaking strategy on actual racing tracks with more opponents is difficult because of the several biases that can influence the overall evaluation (e.g., the car types, the track shape, and also to the driving strategy used by the evaluated controller). In this study, we tried to reduce some of these biases by having only one car model for all the drivers and only one opponent type in each experiment. However, the bias due to the driving policy of the evaluated controller is difficult to eliminate. The implementations of the driving and overtaking behaviors are typically entangled together and it is basically impossible to evaluate different overtaking strategies using the same driving policy. Accordingly, in this study we defined two measures that can mitigate the bias due to the underlying driving policy. Firstly, we defined *BlockedTime* as the total amount of time that a controller spends being blocked by an opponent ahead. More specifically, *BlockedTime* is computed as the number of game tics a controller stays less than 20 m away from an opponent ahead, i.e., when the controller is in a position where it might start or it is performing an overtake maneuver.

An effective overtaking strategy should complete a maneuver as soon as an opponent is close, and thus it should have a low *BlockedTime*. At the same time, it should also have a high number of completed overtaking maneuvers. However, controllers implement different driving policies and for instance a controller might complete more overtakes simply because it is faster than other ones with similar or even better overtaking strategies. Accordingly, we defined *BlockedRatio* as the ratio between the total time spent behind an opponent (*BlockedTime*) and the number of completed overtakes. *BlockedRatio* mitigates the influence of the driving policy because it is normalized with respect to the number of overtakes and it is typically lower for the drivers that spent little time waiting to overtake and completed several maneuvers.[2]

[2] Note that the presence of ten opponents along the track guarantee that the evaluated driver has to overtake quite often in many different section of the track.

Table 7
Tracks used to evaluate the Advanced Overtaking Behavior.



Opponents strategies. In the previous set of experiments, opponents implemented a simple driving policy maintaining their initial speed while driving in the center of the track. In this second set of experiments, opponents implemented a more advanced driving policy and, in each track segment, computed the highest achievable speed; then, while driving, they maintained a speed around the 80% of that value. This driving policy is very safe: indeed opponents will drive slower than they potentially might, however, at the same time they will be able to activate blocking with no danger of going off the track. In particular, an opponent activated the blocking behavior as soon as the distance from the incoming car was less than 75 m behind and implemented one of the three strategies considered in the previous set of experiments (*limited blocking*, *slowly reactive blocking*, and *fully reactive blocking*).

7.2. Experimental results

We compared our controller (*Fuzzy*) and the same five drivers considered in the previous set of experiments (*Berniw*, *Bt*, *Inferno*, *Olethros*, *Simplix*) against opponents implementing the usual blocking strategies, *limited*, *slowly reactive*, and *fully reactive* (see Section 6). The experiments were performed on six tracks from the TORCS distribution that could provide a good mix of shapes, bends, and straight stretches (see Table 7).

Limited blocking. Table 8 reports the average performance of the six drivers against opponents with *limited* blocking on the six tracks considered. In particular, it shows the average *BlockedTime*, the average number of overtaking maneuvers completed, and the average *BlockedRatio*. As can be noted, our controller (*Fuzzy*) has the lowest average *TimeBlocked* with 197.5 tics. *Bt* has the lowest *TimeBlocked* among the other drivers that is however 50% higher than that of *Fuzzy*. *Inferno* and *Olethros* are the drivers with the worse *TimeBlocked*, almost three times larger than our driver. While *Fuzzy* is the fastest to overtake (as it spends the least time following opponents), it performs the least number of overtakes. This is mainly due to the different sensory model that *Fuzzy* uses with respect to the other five drivers. Our controller uses a realistic sensory model (the same used for Simulated Car Racing Championship Loiacono et al., 2009, 2010b) consisting of a belt of laser range finders surrounding the car. This model is quite realistic but at the same time provides poor and noisy information about the surroundings and the track ahead. In contrast, all the other drivers have *exact* information about the environment (the track, the opponents status, etc.) and thus implement rather sophisticated driving policy. For instance, *Simplix* generates the *optimal* racing line for the current track. As a consequence, our driver is much slower than the other ones and, since it needs more time to reach the next opponent, overall it performs fewer overtakes. However, when we consider the average *BlockedRatio*, which is normalized over the number of overtakes completed, we note that *Fuzzy* has the best overall performance, i.e., our driver spend the least time to overtake an opponent (35% less than *Simplix*, the fastest driver).

Slowly reactive blocking. Table 9 reports the average performance of the six drivers against *slowly reactive* opponents on the same tracks (Table 7). As in the previous case, our controller achieves the lowest *BlockedTime* with 186.18 tics while, as before, *Bt* is the second best with a *BlockedTime* of 336.07 tics and the worst one has a *BlockedTime* of 523.10 tics. The number of overtakes against slowly reactive opponents are a little bit less but the overall distribution is similar to the previous case: *Fuzzy* still performs the least number of overtakes. However, if we considere *BlockedTime*, which is normalized with respect to the number of completed maneuvers, *Fuzzy* is still the best performing controller.

Fully reactive blocking. The results of this set of experiments are very much different from the previous cases (Table 10). All the drivers achieved a lower average *BlockedTime* than before and four out of six drivers have a very low average *BlockedRatio* (including our controller, *Fuzzy*). *Olethros* is the best performing driver against *fully reactive* opponents, both in terms of *BlockedTime* and *BlockedRatio*, while in all the previous experiments it performed quite badly. *Fuzzy* has now more overtakes than *Berniw* but in terms of average *BlockedRatio* is slightly worse but it is part of the best performing drivers with a *BlockedRatio* between 4.5 and 6.

The analysis of the racing logs suggest that reacting to incoming vehicles quickly and without avoiding possible collisions (fully reactive blocking is not caution and can take some risk) can be detrimental for opponents which have difficulties to maintain an effective driving strategy while activating/deactivating blocking behaviors. Accordingly, even drivers that demonstrated a rather low performance in all the previous tasks (like *Olethros*) can be competitive against opponents performing fully reactive blocking on complex tracks. It might appear counter-intuitive that opponents with a more advanced blocking strategy are easier to overtake, however, this is coherent with what every player can experience in racing games: there are situations (e.g., while exiting or entering a bend) in which is more convenient to delay the blocking action and keep the current racing line instead of blocking an incoming opponent. Accordingly, it appears that a more cautious blocking strategy (like the slowly reactive one) can be much more effective. At the same time the evaluated controllers fully reactive controllers are more difficult to approach and thus the interaction between evaluated controller and opponents generates the typical rubber band effect (Jimenez, 2009). Accordingly, there is a lower *BlockedTime* since the evaluated controller more often ends at more than 20 m away from the opponent ahead (because of the rubber band effect); at the same time it is more difficult to overtake and thus we also note fewer overtakes with respect to the previous experiments (Table 8 and Table 9).

Overall performance. Table 10 summarizes the drivers' performance over all the blocking behaviors considered in this work. Our controller, *Fuzzy*, demonstrates the best overall performance with the lowest *BlockedTime* (100 tics than the second best, *Bt*) and the lowest *BlockedRatio* (the second best, *Simplix*, has a slight higher *BlockedRatio* but a much higher *BlockedTime*). Thus, the overtaking policy we developed is quite effective as it appears to be the fastest strategy

Table 8
Advanced Overtaking Behavior against opponents with limited blocking.

	Berniw	Bt	Fuzzy	Inferno	Olethros	Simplix
BlockedTime	531.13	327.8	197.5	554.1	476.08	377.25
Completed Overtakes	38.5	36.5	34.5	38.67	41.83	43.5
BlockedRatio	13.8	8.98	5.72	14.33	11.38	8.67

Table 9
Advanced Overtaking Behavior against opponents with slowly reactive blocking.

	Berniw	Bt	Fuzzy	Inferno	Olethros	Simplix
BlockedTime	436.6	331.38	186.18	523.1	518.42	336.07
Completed Overtakes	35.17	42.5	33	36.5	38.5	41.5
BlockedRatio	12.41	7.8	5.64	14.33	13.47	8.1

Table 10
Advanced Overtaking Behavior against opponents with full reactive blocking.

	Berniw	Bt	Fuzzy	Inferno	Olethros	Simplix
BlockedTime	388.02	184.25	161.57	350.83	146.9	165.13
Completed Overtakes	24.5	34.67	29	29.17	31.83	36.17
BlockedRatio	15.84	5.31	5.57	12.03	4.62	4.57

Table 11
Summary of All Behaviors.

	Berniw	Bt	Fuzzy	Inferno	Olethros	Simplix
BlockedTime	451.92	281.14	181.75	476.01	380.47	292.82
Completed Overtakes	32.72	37.89	32.17	34.78	37.39	40.39
BlockedRatio	13.81	7.42	5.65	13.69	10.18	7.25

among the ones analyzed in this work. As already noted, overall our driver completes fewer overtakes since its driving strategy is less effective mainly because of the more realistic but also more limited sensory model. In fact, while all the other drivers can access an exact representation of the track and the opponents' status our driver has only local information about the car surroundings provided by a belt of laser range finders. This greatly influences the driving capabilities of *Fuzzy* that overall is the slowest driver among the ones considered but, as Table 10 shows, the most effecting in overtaking opponents of different types.

8. Conclusions and future research directions

We developed an advanced overtaking behavior that can overtake opponents that implement blocking strategies of increasing difficulty (*limited*, *slowly reactive*, and *fully reactive*) in various racing scenarios. The overtaking was implemented as a fuzzy-system integrated into an existing driver implemented using a fuzzy system that extended the modular architecture developed by one of the authors (Onieva et al., 2009).

We performed two sets of experiments. In the first set of experiments, we considered a simple scenario consisting in a long straight stretch with one blocking opponent to overtake as quickly as possible. In the second set of experiments, we considered a racing scenario involving a variety of tracks and more opponents all implementing a blocking strategy. We compared the performance of our driver against five of the best publicly available drivers for The Open Car Racing Simulator (TORCS).

The experiments we presented show that our fuzzy controller can perform competitively in both scenarios. When racing against one opponent on a single stretch, our fuzzy controller can always complete an overtake with a success rate of 100% while the other drivers we considered show a success rate lower than 40%. In particular, some of the most performing drivers (like *Simplix*) are completely unable to complete an overtake maneuver with the more challenging blocking policies (see Table 5). When racing against more opponents on various track, our controller obtain the best performance over all the blocking strategies (Table 11) both in terms of total time spent blocked by an opponent (*BlockedTime*) and in terms of average time spent blocked by an opponent for each completed overtake (*BlockedRatio*). Noticeably, our simple overtaking behavior shows interesting emerging behaviors resulting into deceiving maneuvers that can entangle the blocking opponent. This suggests that our overtaking strategy can lead to more believable behaviors than the scripted policies implemented in the TORCS drivers.

Our driver works on a realistic sensory model based on an array of laser range finders as opposed to the exact information about the whole race environment (e.g., position on the track and opponent status) available to the other drivers considered in this study. This makes our driver less performing in terms of lap times but makes the approach more general and applicable outside the realm of car racing games. In fact, future research includes, on the gaming side, the addition of a more advanced driving module that should make our driver competitive also in terms of lap times; on the automotive side, we plan to focus on the use of this technology on real cars. Some initial results have been already published in

Milanés, Pérez, Godoy, and Onieva, 2012; Onieva, Godoy, Villagra, Milanés, and Pérez, 2013.

Acknowledgments

This work was supported by the Plan Nacional, under the project Tránsito (TRA2008–06602-C03–01), by the Comisión Interministerial de Ciencia y Tecnología under the project GUIADE (Ministerio de Fomento T9/08) and the Ministerio de Ciencia e Innovación under the project CityElec (PS-370000–2009-4).

References

Butz, M. V. & Lonneker, T. D. (2009). Optimized sensory-motor couplings plus strategy extensions for the torcs car racing challenge. In *IEEE symposium on computational intelligence and games, CIG 2009* (pp. 317–324).

Butz, M. V., Linhardt, M. J., & Lonneker, T. D. (2011). Effective racing on partially observable tracks: Indirectly coupling anticipatory egocentric sensors with motor commands. *IEEE Transactions on Computational Intelligence and AI in Games, 3*(1), 31–42.

Cardamone, L. (2012). Evolutionary learning and search-based content generation in computer games. Ph.D. thesis, Politecnico di Milano.

Cardamone, L., Loiacono, D. & Lanzi, P. L. (2009). On-line neuroevolution applied to the open racing car simulator. In *IEEE Congress on Evolutionary Computation, CEC '09* (pp. 2622–2629).

Cardamone, L., Loiacono, D. & Lanzi, P. L. (2009). Learning drivers for torcs through imitation using supervised methods. In *IEEE symposium on computational intelligence and games, CIG 2009* (pp. 148–155).

Cardamone, L., Loiacono, D. & Lanzi, P. L. (2010). Applying cooperative coevolution to compete in the 2009 torcs endurance world championship. In *2010 IEEE congress on evolutionary computation (CEC)* (pp. 1–8).

Cardamone, L., Loiacono, D., & Lanzi, P. L. (2009c). Evolving competitive car controllers for racing games with neuroevolution. *GECCO '09: Proceedings of the 11th annual conference on genetic and evolutionary computation* (pp. 1179–1186). New York, NY, USA: ACM.

Ebner, M. & Tiede, T. (2009). Evolving driving controllers using genetic programming. In *IEEE symposium on computational intelligence and games, CIG 2009* (pp. 279–286).

Fujii, S., Nakashima, T. & Ishibuchi, H. (2008). A study on constructing fuzzy systems for high-level decision making in a car racing game. In *IEEE international conference on fuzzy systems, FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence)* (pp. 2299–2306).

Garca, R., & de Pedro, T. (1998). Modeling a fuzzy coprocessor and its programming language. *Mathware and Soft Computing, 5*(2–3), 167–174.

Ho, D. T. & Garibaldi, J. M. (2008). A fuzzy approach for the 2007 cig simulated car racing competition. In *IEEE symposium on computational intelligence and games, CIG '08* (pp. 127–134).

Ho, D. T. & Garibaldi, J. M. (2008). A novel fuzzy inferencing methodology for simulated car racing. In *IEEE international conference on fuzzy systems, FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence)* (pp. 1907–1914).

Jimenez, E. (2009). The pure advantage: Advanced racing game AI. <http://www.gamasutra.com/view/feature/132313/thepureadvantageadvanced%.php>.

Kemmerling, M. & Preuss, M. (2010). Automatic adaptation to generated content via car setup optimization in torcs. In *IEEE symposium on computational intelligence and games (CIG)* (pp. 131–138).

Lecchi, S. (2009). Artificial intelligence in racing games. *CIG'09: Proceedings of the 5th International Conference on Computational Intelligence and Games* (pp. 1). Piscataway, NJ, USA: IEEE Press.

Loiacono, D., Togelius, J., Lanzi, P. L., Kinnaird-Heether, L., Lucas, S. M., Simmerson, M., Perez, D., Reynolds, R. G., & Saez, Y. (2008). The wcci 2008 simulated car racing competition. In *IEEE symposium on computational intelligence and games, CIG '08* (pp. 119–126).

Loiacono, D., Cardamone, L. & Lanzi, P. L. (2009). Simulated car racing championship 2009: Competition software manual. Technical Report 2009.04, Dipartimento di Elettronica e Informazione – Politecnico di Milano.

Loiacono, D., Prete, A., Lanzi, P. L. & Cardamone, L. (2010). Learning to overtake in torcs using simple reinforcement learning. In *2010 IEEE world conference on computational intelligence*.

Loiacono, D., Lanzi, P. L., Togelius, J., Onieva, E., Pelta, D. A., Butz, M. V., Lo?nneker, T. D., Cardamone, L., Perez, D., Sa?ez, Y., Preuss, M., & Quadflieg, J. (2010b). The 2009 simulated car racing championship. *IEEE Transactions on Computational Intelligence and AI in Games, 2*(2), 131–147.

Mamdani, E. H. et al. (1974). Application of fuzzy algorithms for control of simple dynamic plant. *Proceedings of the IEE, 121*(12), 1585–1588.

Milanés, V., Pérez, J., Godoy, J., & Onieva, E. (2012). A fuzzy aid rear-end collision warning/avoidance system. *Expert Systems with Applications, 39*(10), 9097–9107.

Munoz, J., Gutierrez, G. & Sanchis, A. (2009). Controller for torcs created by imitation. In *IEEE symposium on computational intelligence and games, CIG 2009* (pp. 271–278).

Onieva, E., Pelta, D. A., Alonso, J., Milanes, V. & Perez, J. (2009). A modular parametric architecture for the torcs racing engine. In *IEEE symposium on computational intelligence and games, CIG 2009* (pp. 256–262).

Onieva, E., Cardamone, L., Loiacono, D. & Lanzi, P. L. (2010). Overtaking opponents with blocking strategies using fuzzy logic. In *IEEE symposium on computational intelligence and games (CIG)* (pp. 123–130).

Onieva, E., Godoy, J., Villagra, J., Milanés, V., & Pérez, J. (2013). On-line learning of a fuzzy controller for a precise vehicle cruise control system. *Expert Systems with Applications, 40*(4), 1046–1053.

Perez, D., Recio, G. & Saez, Y. (2009). Evolving a fuzzy controller for a car racing competition. In *IEEE symposium on computational intelligence and games, CIG 2009* (pp. 263–270).

Preuss, M., Quadflieg, J., & Rudolph, G. (2011). Torcs sensor noise removal and multi-objective track selection for driving style adaptation. In S.-B. Cho, S. M. Lucas, & P. Hingston (Eds.). *CIG* (pp. 337–344). IEEE.

Quadflieg, J., Preuss, M., Kramer, O., & Rudolph, G. (2010). Learning the track and planning ahead in a car racing controller. In G. N. Yannakakis & J. Togelius (Eds.). *CIG* (pp. 395–402). IEEE.

Quadflieg, J., Preuss, M., & Rudolph, G. (2011). Driving faster than a human player. In *EvoApplications*. In C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, J. J. Merelo, F. Neri, M. Preuss, H. Richter, J. Togelius, & G. N. Yannakakis (Eds.). *Lecture Notes in Computer Science* (vol. 6624, pp. 143–152). Springer.

Stanley, K.O. (2004). Efficient evolution of neural networks through complexification. Ph.D. thesis, Department of Computer Sciences, The University of Texas at Austin.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics, 15*(1), 116–132.

The open racing car simulator website. <http://torcs.sourceforge.net/>.

van Hoorn, N., Togelius, J., Wierstra, D. & Schmidhuber, J. (2009). Robust player imitation using multiobjective evolution. In *IEEE congress on evolutionary computation, CEC '09* (pp. 652–659).

Venzon, J. Vdrift: A cross-platform, open source driving simulation. <http://vdrift.net/>.

Wloch, K. & Bentley, P. J. (2004). Optimising the performance of a formula one car using a genetic algorithm. In *Proceedings of 8th international conference on parallel problem solving from nature* (pp. 702–711).

Wolf-Dieter Beelitz, Xavier Bertaux, Brian Gavin, Eckhard M. Jaeger, Kristf Kly-Kullai, Gbor Kmetyk, Enrico Mattea, Jean-Philippe Meuret, Haruna Say, and Andrew Sumner. Speed dreams – a free open motorsport sim and open source racing game. <http://www.speed-dreams.org/>.