Overtaking Opponents with Blocking Strategies Using Fuzzy Logic

Enrique Onieva, Luigi Cardamone, Daniele Loiacono, Pier Luca Lanzi *Member IEEE*

***Abstract*— In car racing, *blocking* refers to maneuvers that can prevent, disturb or possibly 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 advanced blocking strategies. 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 six of the bots available in the TORCS distribution and *simplix*, a state-of-the-art bot which won the 2009 TORCS Endurance World Championship. The comparison was carried out against opponents implementing three blocking strategies of increasing difficulty. The results we present show that our strategy can overtake the opponent car in all the considered scenarios. In contrast, all the other bots can complete the overtaking maneuvers in only less than 40% of the cases. Our strategy is slightly more risky than others and may result in limited rear and lateral damage. Other more cautious drivers receive almost no damage, however they can overtake only around 30% of the cases.**

I. Introduction

Overtaking plays a key role in the development of the artificial intelligence (AI) for racing games. Overtaking skills are essential for the development of successful controllers [1] but they are also extremely important in terms of AI believability and players' satisfaction. Alas, overtaking is very complex and difficult to program in that it requires the cooperative management of all the vehicle actuators and has to deal with a large variety of unexpected, challenging, and dangerous situations. An analysis of the most competitive bots available for TORCS [2] (a state-of-the-art open source car racing simulator) shows that even the most advanced drivers (e.g., the winner of the the 2009 TORCS Endurance World Championship) might fail in dealing with simple but very common scenarios, such as those involving opponents with blocking capabilities.

Blocking is a rather typical maneuver in racing games that is frequently employed both by human players and non-playing characters. The term identifies all those strategies that a driver can use to prevent, disturb or possibly block an overtaking action by an incoming car. Blocking strategies are typically implemented by adapting the vehicle trajectory so as to block the racing line chosen by the incoming vehicle. An analysis we performed on seven of the most competitive drivers available for TORCS [2] shows that while most of these drivers implement reliable overtaking strategies, their performance is dramatically reduced when facing opponents implementing even simple blocking strategies.

In this work, we studied the development of an advanced overtaking behavior, implemented using a fuzzy system, that can overtake challenging opponents implementing blocking strategies of increasing difficulty. In particular, we considered 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. We compared a simple driver using our overtaking behavior against seven of the best drivers available for The Open Car Racing Simulator (TORCS) [2] in terms of percentage of overtakes successfully completed; percentage of *rear damage* and *lateral damage* suffered by the overtaking car; and the time to complete the maneuver. Our results show that our overtaking strategy can successfully overtake a blocking opponent. In contrast, all the other bots can complete the overtaking maneuvers in only less than 40% of the cases. Our strategy is slightly more risky than the ones used by other drivers and may result in some rear and lateral damage. However, more cautious drivers, which receive almost no damage, can overtake only in around 30% of the cases.

II. Related Work

In the recent years, several works have been focusing on car racing games [3], [4], [1], [5], [6], [7], [8], [9], [10], [11], [12], [13]. The range of techniques applied in this area covers the entire Computational Intelligence field and includes neural networks [1], [4], fuzzy [9], [10], [11], [12], evolutionary algorithms [3], [6], supervised learning [5], [7], [8], etc.

There are however only few works with a specific focus on the study of overtaking behaviors. Cardamone et al. [1] applied NEAT to evolve a neural network that can overtake in different parts of a track. The overtaking behavior was activated on top of a main driving behavior each time an opponent was closer above a defined threshold. Loiacono et al.[14] applied reinforcement learning to learn two overtaking behaviors: (i) overtaking on a straight exploiting the drag effect of the opponent; (ii) overtaking close to a turn using braking delay. Recently, Onieva et al. [9] introduced a modular architecture to develop a complete racing behavior

E. Onieva is in the Industrial Computer Science Department, Centro de Automática y Robótica (UPM-CSIC). La Poveda-Arganda del Rey, 28500 Madrid, Spain. email: enrique.onieva@car.upm-csic.es Luigi Cardamone (cardamone@elet.polimi.it), Daniele Loiacono (loiacono@elet.polimi.it) and Pier Luca Lanzi (lanzi@elet.polimi.it) are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy. Pier Luca Lanzi (lanzi@illigal.ge.uiuc.edu) is also member of the Illinois Genetic Algorithm Laboratory (IlliGAL), University of Illinois at Urbana Champaign, Urbana, IL 61801, USA.

978-1-4244-6297-1/10/$26.00 ©2010 IEEE

by means of a fuzzy system. The architecture includes a specific module that implements the overtaking behavior by adjusting or replacing the outputs of the driving modules in presence of opponents.

III. Blocking Behavior

Overtaking is a complex maneuver whose success heavily depends on the behavior of the opponents which can react in different ways in order to prevent the overtake.

A rather typical overtaking situation involves an opponent that performs a blocking maneuver to prevent being overtaken: as soon as the opponent is approached by the overtaker, the former modifies his direction to block the racing line followed by the latter. In this work, we tackle the development of an overtaking behavior capable of dealing with such a complex scenario. In particular, we considered opponents implementing three blocking strategies of increasing difficulty: *limited*, *slowly reactive*, and *fully reactive*. Drivers implementing a *limited* blocking strategy mimic the trajectory followed by the overtaker but, to avoid possible collisions, they 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. 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. 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.

IV. System Architecture

Figure 1 provides an overview of the architecture of the proposed system. It is based on a classical three-layers hierarchy architecture: a perception layer where the input signals are acquired from the sensors, a control layer where actions are taken based on the input signals and an actuation layer that acts upon the vehicle control elements.

The perception layer receives positions and speeds of the cars from TORCS engine. The decision layer is composed by a fuzzy system, explained in detail in Section V, that computes the target speed and the target deviation. These values are sent to the actuation layer, composed of three simple modules, each of which controls gear, pedals (throttle and brake unified) and steering.

The decision layer is based on a computational model of a fuzzy co-processor named ORBEX (Spanish acronym for *Fuzzy Experimental Computer*)[15]. Driving strategies can be defined and implemented by means of *if ... then ...* rules in a quasi-natural language, for instance:

If *speed_error* **more than** null **then** *throttle* up

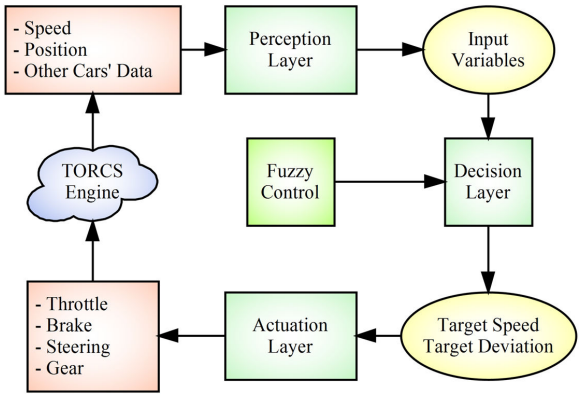

Fig. 1. Overview of the system architecture.

where the terms in italic are fuzzy variables, the terms in bold are ORBEX language keywords, and the terms in plain script are linguistic variable values. The terms in the first part of the rule (i.e., before the keyword **then**) are input variables, while terms on the right part are output variables. The inference engine performs the fuzzy computations and assigns singleton values to the output variables.

The t-norm *minimum* and the t-conorm *maximum* are used to implement *and* and *or* operators. Mamdani-type inference [16] is used, and the defuzzification operator is the center of mass. In the system, input membership function are shaped with trapezoids and output membership functions with singletons [17]. Therefore, the crisp value of y_{out} of an output variable is calculated as shown in (1).

$$y_{out} = \frac{\sum y_i \cdot w_i}{\sum w_i} \tag{1}$$

where w_i represents the weight of the i-th rule and y_i is the value of the output y inferred by the i-th rule.

Sugeno [18] proved that fuzzy systems modeled with singleton consequents are a special case of systems modeled with trapezoidal consequents and that they have the same expressive power. In particular, Sugeno suggested that trapezoidal consequents are necessary only to use fuzzy terms in the consequents of the rules, which is not the case in this work. Singletons based fuzzy systems are very commonly used in practical control system applications [19], [20], [21], [22] because they allow fast calculations and as well as an easy and understandable design of controllers.

V. Fuzzy Controller Design

This section explains in detail the architecture of the fuzzy controller. The controller, and in particular rules and membership functions, has been designed by a human expert. In particular, the parameters used to encode rules and membership functions were not fully optimized, but they were designed to keep the controller as simple as possible to understand and to represent. In addition, rules were designed to be reliable with respect to small changes in the membership functions as well as to possible errors in the sensors readings.

A. Input and Output Variables

Let (x_o, y_o) and (x_b, y_b) be respectively the positions of the *overtaker* and of the *blocking* opponent, where x denotes the distance from the start line along the track and y the deviation with respect to the center of the track. It is defined D_x as the distance between both vehicles along the track $(x_o - x_b)$. Taking into account lengths of the vehicles $(l_o = l_b = len)$, three linguistic situations are described: (i) if $(D_x < -len)$ the overtaker has passed the opponent and the maneuver can be finalized without risk; (ii) if $(-len < D_x < len)$ both vehicles are circulating in parallel; (iii) if $(D_x > len)$ the overtaker is behind the opponent. Three trapezoidal membership functions are used to codify D_x, they are shown in figure 2.

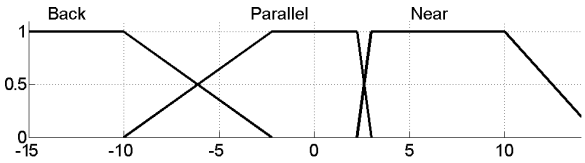

Fig. 2. Membership function used to codify D_x.

The speed difference between both vehicles ($diff_{speed} = (\dot{x}_1 - \dot{x}_2)$) in m/s is used to calculate the variable *Time to Collision* ($Tt_C = D_x/diff_{speed}$), that represents the remaining time in seconds until both cars are in the same x. This variable is codified by four membership functions as shown in figure 3.

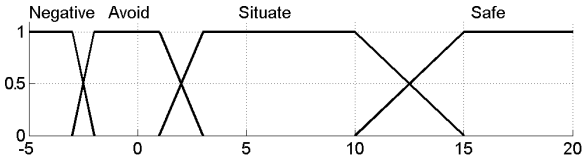

Fig. 3. Membership function used to codify Tt_C.

The distance D_y represents the lateral position of the overtaker vehicle with respect to the *opponent* $(y_b - y_o)$, in meters. Depending on the value of D_y and the vehicle's width $(w_o = w_b = wid)$, we can define five cases:

- If $(D_y < -wid)$ then the blocking car is in the right part from the overtaker without crash possibility.
- If $(-wid < D_y < 0)$ then the blocking car is in the right, but both cars can crash.
- If $(D_y = 0)$ then the opponent is just behind or in front.
- If $(0 < D_y < wid)$ then the blocking car is on the left with crash risk.
- If $(wid < D_y)$ then the blocking car is on the left, without crash risk.

D_y is codified using the five membership functions as shown in Figure 4.

Current deviation (C_D) of the overtaker is also considered with the aim of knowing the free space by each side. This position is normalized with respect to the track width, so $(C_D = 0)$ means that the vehicle is in the center of the road,

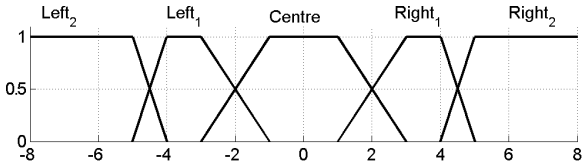

Fig. 4. Membership function used to codify D_y.

$(C_D < 0)$ that it is in the left part of the road and vice versa. To codify this variable has been used eight functions as shown in figure 5.

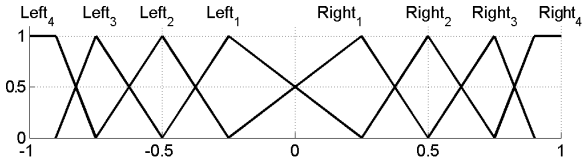

Fig. 5. Membership function used to codify C_D.

The fuzzy system calculates target deviation (T_D) values to send to a lower level layer in charge of acting over the steering wheel; this values are normalized with respect to the track width $(T_D \in [-1, 1])$. Nine singletons codify this output variable, they are shown in figure 6 (top). Emergency pedals actions (E_P) is codified with two singletons as shown in figure 6 (bottom); they represent respectively, to act with maximum value over the throttle and brake, in order to perform an emergency acceleration or deceleration. While this value is zero, the car will manage pedals like driving alone.

Fig. 6. Membership functions used to codify T_D (top) and E_P (bottom).

B. Rule Base

Rule base is built by means of the four input variables (D_x, Tt_C, D_y, and C_D) presented previously.

It is important to note that in case ($D_x = Back$ or $Tt_C = Safe$) no overtaking actions must be performed because the opponent is either behind or too far.

In case $(Tt_C = Situate)$ the driver aims to reach a safe lateral position suitable to start the overtaking maneuver. In case the opponent is moving toward one border of the track, the rules will lead to attempt the overtaking maneuver on the opposite border of the track. Rules implemented in this case

are shown in Table I, where columns represent the variable C_D and the rows the variable D_y. No pedals actions are performed in this case.

TABLE I
T_D FUZZY RULES IN CASE $Tt_C = Situate$.

C_D / D_y	$Left_2$	$Left_1$	$Center$	$Right_1$	$Right_2$
$Left_4$	$Left_4$	$Left_3$	$Left_3$	$Center$	$Left_4$
$Left_3$	$Left_3$	$Left_2$	$Left_2$	$Center$	$Left_4$
$Left_2$	$Left_2$	$Left_2$	$Left_1$	$Center$	$Left_3$
$Left_1$	$Left_1$	$Center$	$Right_1$	$Left_2$	$Left_2$
$Right_1$	$Right_2$	$Right_2$	$Right_1$	$Center$	$Right_1$
$Right_2$	$Right_3$	$Center$	$Right_1$	$Right_1$	$Right_2$
$Right_3$	$Right_4$	$Center$	$Right_2$	$Right_2$	$Right_3$
$Right_4$	$Right_4$	$Center$	$Right_3$	$Right_3$	$Right_4$

When ($Tt_C = Avoid$) the two vehicles are too close. If ($D_y = Left_2$ *or* $Right_2$) both vehicles are separated enough to perform the overtake without risk and, the current C_D value will be maintained with no pedals actions (E_P). This is obtained by eight rules generated as:

IF($Tt_C = Avoid$)AND($D_y = Left_2$OR$D_y = Right_2$)
AND($C_D = TAG$)THEN($T_D = TAG$),

where $TAG = (Left_{1,2,3,4} \cup Right_{1,2,3,4})$.

Rules for case ($Tt_C = Avoid$ *and* $D_x = Near$) are reported in Table II. These rules will allow the vehicle to overtake in a proper way, increasing or decreasing the current speed when necessary.

TABLE II
T_D AND E_P FUZZY RULES IN CASE $Tt_C = Avoid$ AND $D_x = Near$.

C_D / D_y	$Left_1$	$Center$	$Right_1$
$Left_4$	$Center$ + $Slow$	$Center$ + $Slow$	$Right_1$ + $Fast$
$Left_3$	$Center$ + $Slow$	$Center$ + $Slow$	$Right_1$ + $Fast$
$Left_2$	$Left_4$ + $Fast$	$Right_4$ + $Slow$	$Right_1$ + $Fast$
$Left_1$	$Left_4$ + $Fast$	$Right_4$	$Right_1$ + $Fast$
$Right_1$	$Left_1$ + $Fast$	$Right_4$	$Right_4$ + $Fast$
$Right_2$	$Left_1$ + $Fast$	$Left_2$ + $Slow$	$Right_4$ + $Fast$
$Right_3$	$Left_1$ + $Fast$	$Center$ + $Slow$	$Center$ + $Slow$
$Right_4$	$Left_1$ + $Fast$	$Center$ + $Slow$	$Center$ + $Slow$

In order to finish the maneuver cautiously, once both vehicles are driving parallels, when the opponent is on the right side, rules lead the overtaker very close to left border, and vice versa. This is implemented by the following rules:

- IF $D_x = Parallel$ AND ($D_y = Right_1$ OR $D_y = Right_2$) THEN $T_D = Left_4$
- IF $D_x = Parallel$ AND ($D_y = Left_1$ OR $D_y = Left_2$) THEN $T_D = Right_4$

Additionally, an emergency braking action is carried out when the blocking car is just in front and very close, according to the following rule: IF ($D_x = Near$ AND $D_y = Center$) THEN $E_P = Slow$.

VI. Design of Experiments

We performed a set of experiments to compare our overtaking behavior against six of the best bots distributed with TORCS and *simplix*, a state-of-the-art bot which won the 2009 TORCS Endurance World Championship. In particular, all the bots considered in this paper have been specifically developed for TORCS and implement (following slightly different approaches) the typical controller architecture used also in commercial racing game [23], i.e., searching for the best racing line to follow on the track. All the experiments were performed on a straight stretch, 15m wide and 2800m long; all the drivers used the same car model *car1-trb1* available in the TORCS distribution. Each experiment was divided into a set of trials each one involving two cars: one controlled by an overtaking bot (the *overtaker*); one controlled by an *opponent* driver implementing one of the three blocking strategies we considered (Section III). At the beginning of each trial, both cars were positioned in the center of the track; the overtaking car was d meters behind the opponent ($d \in \{80, 100, 120, \ldots, 300\}$). Both cars had an initial speed of 170km/h; the opponent was programmed to maintain the initial speed but was allowed to change its trajectory to block the incoming vehicle as soon as the distance between the two cars was less than 75m. Each trial runs until either (i) the opponent reached the end of the straight without having been overtaken; or (ii) the overtaking car has successfully completed the maneuver and was 50m ahead of the opponent. Thus, each experiment consisted of 96 trials, i.e., 12 trials (one for each value of d) for 8 possible overtaking drivers.

During each experiment, for each driver, we measured the percentage of overtakings successfully completed; the percentage of *rear damage* and *lateral damage* suffered during the maneuvers; and the time to complete the overtakings.

TABLE III
OVERTAKING AN OPPONENT WITH THE *limited* BEHAVIOR.

	fuzzy	*berniw*	*bt*	*inferno lliaw*	*olethros*	*simplix*	*tita*
$\%S$	100	0	0	0	0	100	0
$\%B_D^f$	91.7	100	100	100	100	100	100
$\%L_D^f$	100	100	100	100	100	100	100

VII. Experimental Results

In this section, we discuss the results of the experiments we performed to evaluate our overtaking strategy. Our analysis takes into account seven bots (*berniw*, *bt*, *inferno*, *lliaw*, *Olethros*, *Tita*, and *simplix*) and three opponents implementing the limited, slowly reactive, and fully reactive blocking strategies.

A. Limited Blocking

An opponent with a *limited* blocking strategy tries to make overtaking difficult but it does not block the overtaking action completely. For this purpose, it adapts its trajectory to the one followed by the incoming car (the overtaker) but it avoids going too close (i.e., less than 4.5m) to the track borders.

Table III reports the performance of our controller (reported as **fuzzy**) against the other seven bots considered

in this work; $\%S$ represents the percentage of successful overtakes performed; $\%B_D^f$ represents the percentage of experiments carried out without rear damage; $\%L_D^f$ represents the percentage of experiments carried out without lateral damage. Although this blocking behavior is pretty simple and allows for the completion of overtaking actions, all the bots available with the TORCS distribution fail to complete any overtake. Only our controller (**fuzzy**) and *simplix*, one of the best controllers available for TORCS, completed all the overtakes. All the other controllers remained in the back following the blocking vehicle, without crashing with it. None of the controllers we considered received any lateral damage. Our controller, **fuzzy**, received some limited lateral damage only in one out of the 12 experiments preformed, namely when $d = 120$.

Figure 7 shows the maximum difference between the speed of the two vehicles (the overtaker and the opponent) before the overtake occurs (note that, *Inferno*, *Lliaw* and *Tita* have exactly the same speed difference, so their lines are superimposed). The reported speed differences ranges from 50 km/h, when the initial distance is 80 meters, to more than 100 km/h, when the initial distance is 300m. Accordingly, all the bots considered are in principle capable to overtake the opponent, in that they reach it with enough speed difference to perform an overtaking maneuver.

It is interesting to analyze the overtaking strategies employed by **fuzzy** and *simplix* to reach a 100% overtaking success rate. As an example, Figure 8 compares the trajectories of our controller (upper) and *simplix* (lower) on 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 fuzzy controller performs a more advanced maneuver to get around the blocking. At the beginning, until time-stamp 8, **fuzzy** performs the typical overtaking maneuver: finding that the opponent is on its trajectory it changes direction and starts the overtaking. As **fuzzy** is getting close, 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, **fuzzy** implements a rather advanced and realistic strategy. In fact, it almost appears as our bot performed a small deceiving maneuver that basically entangled the opponent during the second wider direction change.

Figure 9 compares the time **fuzzy** and *simplix* required to complete the overtake maneuver for each initial setup; in most of the cases, *simplix* is slightly faster than our controller. Our analysis suggests that such a difference may be due to the difference between the approaching speed of the two drivers (see Figure 7); the only exception is the case with an distance of 120m, where **fuzzy** slightly hits the opponent's rear so as to slow down the overtaking action.

TABLE IV
OVERTAKING AN OPPONENT WITH THE *slowly reactive* BEHAVIOR.

	fuzzy	*berniw*	*bt*	*inferno lliaw*	*olethros*	*simplix*	*tita*
$\%S$	100	91.7	58.3	100	83.3	0	100
$\%B_D^f$	91.7	33.3	66.7	41.7	100	100	50
$\%L_D^f$	83.3	0	91.7	8.3	83.3	100	8.3

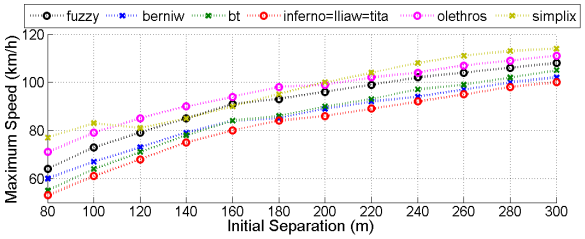

Fig. 7. Maximum speed difference against the *limited* opponent.

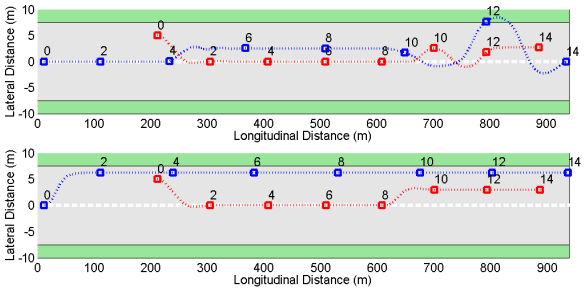

Fig. 8. Trajectories of the fuzzy controller (top) and *simplix* (bottom) against the *limited* opponent and initial separation of 200m.

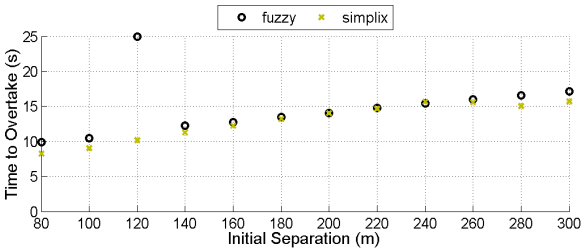

Fig. 9. Time to overtake against the *limited* opponent.

B. Slowly Reactive Blocking

An opponent with a slowly reactive blocking strategy adapts its trajectory to block the overtaking bot with a delay of one second; in contrast to the previous case, now the opponent has no limitations and can drive very close to the track borders so as to block the overtaking driver completely.

Table IV reports the performance of the bot using our overtaking controller (reported as **fuzzy**) against the other seven bots considered in this analysis. 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 severe damage to most of the drivers. In fact, only *olethros* and *simplix* received no rear damage while only *simplix* received no lateral damage. With this setup, *simplix* cannot complete any overtake because of its very cautious driving policy (in fact, it does not receive any damage). Overall, it appears as our controller **fuzzy** may provide the best trade-off as it can complete all the overtaking maneuvers while receiving limited damage. Figure 10 reports the time to complete the overtaking for each possible configuration (no data point is reported for drivers that did not complete the overtaking). As can be noted, our controller (**fuzzy**) and *olethros* implement the fastest overtaking strategies. *bt* performs similarly to **fuzzy** and *olethros* only when the initial distance is greater than 180m. All the other drivers are generally much slower. Figure 11 compares the overtaking behaviors of **fuzzy**, *olethros*, and *berniw* against the slowly adaptive opponent when the initial distance is 180m. As can be noted, our controller (top) completes the overtaking in the lesser time while keeping a rather safe 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 receives significant damage during the maneuver (a 30% lateral damage and a 15% rear damage).

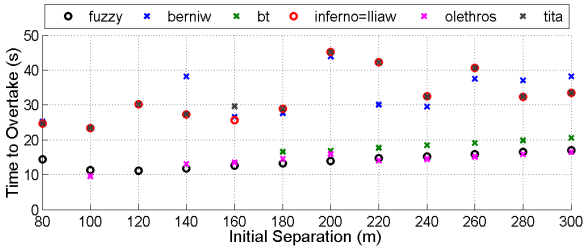

Fig. 10. Time to overtake against the *slowly reactive* opponent.

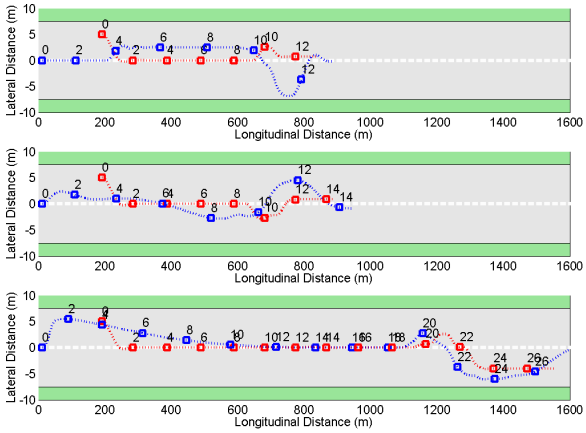

Fig. 11. Trajectories of the *fuzzy* (top), *olethros* (center) and *berniw* (bottom) against the *slowly reactive* opponent and initial separation of 180m.

C. Fully Reactive Blocking

An opponent following a *fully reactive blocking* strategy rapidly adapts its trajectory to block the incoming overtaking car. Compared to the previous cases, this type of opponent is very challenging in that it can completely block an overtaking action and can do it very quickly.

TABLE V
OVERTAKING AN OPPONENT WITH THE *fully reactive* BEHAVIOR.

	fuzzy	*berniw*	*bt*	*inferno lliaw*	*olethros*	*simplix*	*tita*
$\%S$	100	0	0	0	0	0	0
$\%B_D^f$	91.7	100	100	100	100	100	100
$\%L_D^f$	83.3	100	100	100	100	100	100

Table V reports the performance of all the bots we considered. As can be noted, our controller (**fuzzy**) is the only one capable of completing an overtaking maneuver for each setup, while causing some rear damage only in one case. In contrast, all the other controllers (even the very advanced *simplix*) cannot complete an overtaking not even once. Although the opponent presents a very challenging behavior, the **fuzzy** controller overtakes in all the cases, and produced a rear damage only in one case.

As an example, Figure 12 compares the trajectory followed by our **fuzzy** controller (first from top), *olethros* (second from top), *berniw* (third) and *simplix* (bottom). Our fuzzy controller has a completely different and highly realistic behavior when compared to the bots distributed with TORCS (*olethros* and *berniw*) and the very competitive *simplix*. As in other cases, 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.

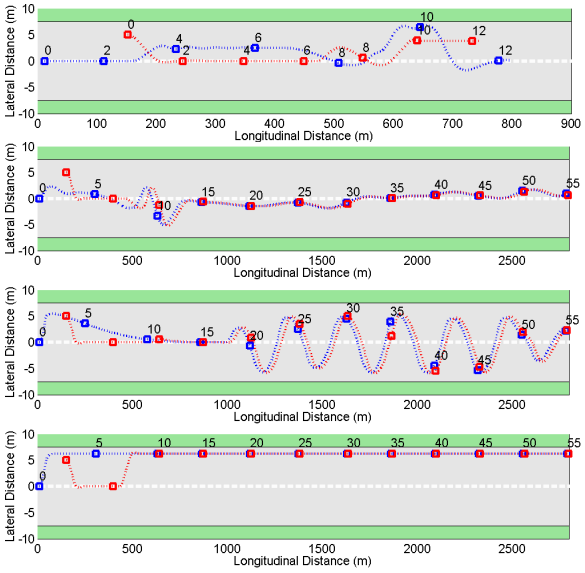

Fig. 12. Trajectories of **fuzzy** (first from top) *olethros* (second from top), *berniw* (third) and *simplix* (bottom) against the *fully reactive* opponent and initial separation of 140m.

TABLE VI
OVERALL OVERTAKING PERFORMANCE.

	fuzzy	*berniw*	*bt*	*inferno lliaw*	*olethros*	*simplix*	*tita*
$\%S$	100	34.4	21.9	37.5	31.3	25	37.5
$\%B_D^f$	90.6	75	87.5	78.1	100	100	81.3
$\%L_D^f$	87.5	62.5	96.9	65.6	93.8	100	65.6

D. Discussion

Table VI summarizes the results of the three sets of experiments; for each driver, the table reports the average performance achieved by each controller against the three blocking strategies. As can be noted, only our **fuzzy** controller can overtake in all the configurations considered reaching 100% performance ($\%S$). In contrast, all the other bots can successfully overtake only in less than 40% of the cases. This low performance is either due to a cautious driving policy (this is the case of *simplix* and *olethros* which receive almost no rear nor lateral damage) or to the inability to deceive the blocking strategy. Overall, our **fuzzy** controller takes some risks which result in limited rear and lateral damage but can complete all the overtakes so as to provide the best trade-off between the overtaking capabilities and the damage suffered.

VIII. CONCLUSIONS

In this work, we developed an advanced overtaking behavior that can deal with complex scenarios involving opponents with blocking capabilities. We considered opponents implementing three blocking strategies of increasing difficulty (*limited*, *slowly reactive*, and *fully reactive*). The behavior was implemented as a fuzzy-system integrated into an existing driver. Our driver was then compared to seven of the best drivers available for The Open Car Racing Simulator in terms of percentage of overtakes successfully completed, percentage of *rear damage* and *lateral damage* suffered by the overtaking car, and the time to complete the maneuver. The results we presented show that our system can overtake in all the scenarios we considered; in contrast, all the other bots (even the most advanced ones) complete less than the 40% of maneuvers. Noticeably, our simple overtaking behavior demonstrated some interesting emerging behavior resulting into deceiving maneuvers that can entangle the blocking opponent. Our study is preliminary as it only considers overtaking on a straight stretch. However, our study showed that existing overtaking strategies (even the ones implemented by rather advanced drivers) are extremely limited and that can be outperformed even by a simple set of fuzzy-rules (as the ones we used).

Future research directions include more complex overtaking tasks, such as overtaking before or during a bend, as well as more realistic scenarios, such as overtaking in a two-way road. In addition, future research will focus on adapting dynamically the overtaking behavior to the current race scenario, e.g., exploiting a careful behavior when the car is damaged or the driver is leading the race.

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

[1] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Evolving competitive car controllers for racing games with neuroevolution," in *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2009, pp. 1179–1186.

[2] "The open racing car simulator website." [Online]. Available: http://torcs.sourceforge.net/

[3] M. V. Butz and T. D. Lonneker, "Optimized sensory-motor couplings plus strategy extensions for the torcs car racing challenge," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 317–324.

[4] L. Cardamone, D. Loiacono, and P. Lanzi, "On-line neuroevolution applied to the open racing car simulator," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, May 2009, pp. 2622–2629.

[5] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Learning drivers for torcs through imitation using supervised methods," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 148–155.

[6] M. Ebner and T. Tiede, "Evolving driving controllers using genetic programming," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 279–286.

[7] N. van Hoorn, J. Togelius, D. Wierstra, and J. Schmidhuber, "Robust player imitation using multiobjective evolution," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, May 2009, pp. 652–659.

[8] J. Munoz, G. Gutierrez, and A. Sanchis, "Controller for torcs created by imitation," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 271–278.

[9] E. Onieva, D. A. Pelta, J. Alonso, V. Milanes, and J. Perez, "A modular parametric architecture for the torcs racing engine," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 256–262.

[10] D. Perez, G. Recio, and Y. Saez, "Evolving a fuzzy controller for a car racing competition," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 263–270.

[11] S. Fujii, T. Nakashima, and H. Ishibuchi, "A study on constructing fuzzy systems for high-level decision making in a car racing game," June 2008, pp. 3626–3633.

[12] D. Ho and J. Garibaldi, "A fuzzy approach for the 2007 cig simulated car racing competition," in *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, Dec. 2008, pp. 127–134.

[13] D. Loiacono, J. Togelius, P. Lanzi, L. Kinnaird-Heether, S. Lucas, M. Simmerson, D. Perez, R. Reynolds, and Y. Saez, "The wcci 2008 simulated car racing competition," in *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, Dec. 2008, pp. 119–126.

[14] D. Loiacono, A. Prete, P. L. Lanzi, and L. Cardamone, "Learning to overtake in torcs using simple reinforcement learning," in *2010 IEEE World Conference on Computational Intelligence*, July 2010.
[15] R. Garca and T. de Pedro, "Modeling a fuzzy coprocessor and its programming language," *Mathware and Soft Computing*, vol. 5(2-3), pp. 167–174, 1998.
[16] E. Mamdani *et al.*, "Application of fuzzy algorithms for control of simple dynamic plant," *Proc. Iee*, vol. 121, no. 12, pp. 1585–1588, 1974.
[17] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE transactions on systems, man, and cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.
[18] M. Sugeno, "On stability of fuzzy systems expressed by fuzzy rules withsingleton consequents," *IEEE Transactions on Fuzzy systems*, vol. 7, no. 2, pp. 201–224, 1999.
[19] C. F. Juang, C. T. Chiou, and C.-L. Lai, "Hierarchical singleton-type recurrent neural fuzzy networks for noisy speech recognition," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 833–843, 2007.
[20] E. Jahanshahi, K. Salahshoor, and Y. Sahraie, "Application of fuzzy observer and controller in gas-lifted oil wells," in *Proc. IEEE International Conference on Networking, Sensing and Control ICNSC 2008*, 2008, pp. 101–106.
[21] L. L. Simon and K. Hungerbuehler, "Real time takagi-sugeno fuzzy model based pattern recognition in the batch chemical industry," in *Proc. (IEEE World Congress on Computational Intelligence). IEEE International Conference on Fuzzy Systems FUZZ-IEEE 2008*, 2008, pp. 779–782.
[22] S. Jinju, W. Minxiang, and W. Weidong, "Robust takagi-sugeno fuzzy control for a mini aviation engine," in *Proc. 27th Chinese Control Conference CCC 2008*, 2008, pp. 775–780.
[23] S. Lecchi, "Artificial intelligence in racing games," in *CIG'09: Proceedings of the 5th international conference on Computational Intelligence and Games*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1–1.