Evolving a rule system controller for automatic driving in a car racing competition

Diego Perez, Yago Saez *Member, IEEE, Gustavo Recio, Pedro Isasi*

Abstract— **The techniques and the technologies supporting Automatic Vehicle Guidance are important issues. Automobile manufacturers view automatic driving as a very interesting product with motivating key features which allow improvement of the car safety, reduction in emission or fuel consumption or optimization of driver comfort during long journeys. Car racing is an active research field where new advances in aerodynamics, consumption and engine power are critical each season. Our proposal is to research how evolutionary computation techniques can help in this field. For this work we have designed an automatic controller that learns rules with a genetic algorithm. This paper is a report of the results obtained by this controller during the car racing competition held in Hong Kong during the IEEE World Congress on Computational Intelligence (WCCI 2008).**

I. Introduction

In 1970, the visionary Robert E. Fenton, in his survey, predicted how automatic vehicle guidance would evolve, [1]:

The system would probably be implemented in three overlapping stages. The first could be the installation and use of various driver aids so that the driver would be a more effective decision maker and improve the performance of the driver-vehicle system. The second stage could involve the gradual introduction of various subsystems for partial automatic control. The third would include the transition to complete automatic vehicle control. Each of these stages must be realized within the confines of one system so that the addition of each feature would contribute to the ultimate system.

Today, we are in the second stage: we can buy cars with electronic aided brake systems (ABS), lane change aids, adaptive cruise controls which maintain a set distance from the car ahead, automatically accelerating or decelerating, and even applying the brakes, etc. All these new technical advances are a good starting point for approximating the third stage: to complete automatic vehicle control.

The techniques and the technologies supporting Automatic Vehicle Guidance (AVG) are an important issue [2]. Automobile manufacturers view automatic driving as a very interesting product with motivating key features which allow improving of the car safety, reduction in emission or fuel consumption or optimizing of the driver comfort during long journeys.

This topic has been addressed by numerous researchers, e.g. [3], [4], [5], [6], [7], [8], as an engineering problem. However, this problem can also involve, among others, robotics, artificial intelligence, computer engineering, telecommunications, signal and image processing, or control and automation techniques in order to develop efficient systems for automatic driving.

Many different approaches have been studied in recent years and the most promising ones are being engineered on real prototypes, i.e., [6], [9] in the ARGO project, the Buick from the California PATH project or [10]. This is due to the fact that today more technological possibilities allow development of completely functional prototypes with lower costs.

In fact, there is a well-known annual competition organized by the Defense Advanced Research Projects Agency (DARPA) for driverless cars which evaluates all the international improvements carried out annually in this field. An example can be found in [11]. However, researchers have a long way to go until these intelligent vehicles capable of driving in a fully automated way are actually available. Most of the works mentioned before was engineered in real prototypes, this involves two main problems: the costs of buying and modifying a car, and the time needed for each test. To overcome these constraints we propose to use car simulators. Today, car simulators are very close to reality, and they allow us to speed up the research by testing more techniques.

A. Evolutionary computation techniques applied to the automated driving

Our proposal for this task is to research how evolutionary computation techniques can help in automatic driving. For this proposal we have collected some work related to evolutionary computation techniques applied to automated driving.

These works fall into two different categories: autonomous driving and vehicle features optimization.

1) Autonomous driving: one of the first approaches in this area was carried out at the Carnegie Mellon University by Sukthankar *et al.* in 1996, [7]. This work uses reasoning modules which combine high level task goals with low-level sensor constraints. Those modules are directly dependent on a large number of parameters, and the setting of these parameters must be done carefully. As this manual selection is tedious and error prone, a Population Based Incremental Learning (PBIL), which is a combination of Genetic Algorithms (GAs) and competitive learning, [12], is proposed for automatically setting each module's parameters.

Therefore, the algorithm will propose what the probabilities of using a set of rules are depending on each situation. The evaluation function takes into account different aspects,

The authors are with Carlos III University of Madrid, Av. de la Universidad 30, Madrid, Spain (contact phone: +34-91-624-8456; email: `yago.saez@uc3m.es`).

such as serious crashes, collisions, wrong exits, distance completed, etc. For the simulation, the system uses a program called SHIVA (Simulated Highways for Intelligent Vehicle Algorithms) which reproduces a microsimulation of vehicles moving and interacting in a user-defined roadway. The algorithm can influence the vehicles' motion sending simulated commands (steering, accelerate and brake). In addition, the system provides a perception module responsible for obstacle detection, positioning, lane tracking, vehicle sensing, etc. Two years later the PBIL was compared to the GA in this same framework, [13], see Figure 1. The results of these works were quite motivating; at the end of experiments the vehicles automatically entered the test track, completed one lap and a half and finally took the exit. Although they were not capable of avoiding collisions with other vehicles, these experiments showed the potential for intelligence behavior in the tactical driving.

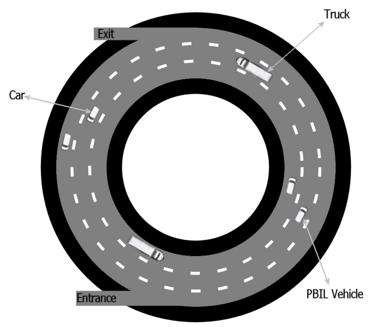

Fig. 1. An example of the Cyclotron test track.

Other interesting work carried out in 1996 by Pyeatt *et al.* from the Colorado University deals with simulated race car driving [14]. In this case a study on autonomous driving was developed based upon RARS simulator software (also known as TORCS). This simulator gives information about the vehicle position, vehicle speed, distance to the current track segment, curvature of the track, and relative position and velocity of nearby cars. It also offers the possibility of controlling the speed and steering of the vehicle.

This work applied neural networks for learning automatic driving through the use of input such as position, speed, distance measured till the end of the segment, angle between the vehicle and the road, etc. This neural network produced a set of rules which decided when to accelerate, brake or steer. The results showed that the RARS simulator was adequate for developing the test framework and that neural networks can be competitive techniques for producing autonomous racing cars.

In 1998, Bernard *et al.* from the Iowa State University illustrated the power of GAs to model driver/vehicle behavior. In fact, their work determined how fast and safe a given vehicle model could be driven through a short course without failure from hitting a cone or lifting the wheels. In this work they used a simulator which they carried out. The GA represents the vehicle movements with the starting and ending points of the path, and a measure of the position with the first and second derivatives of the path in a point near the middle set of cones. The results were very good but they only reached a near optimal solution due to the constraints of the genetic representation selected.

In 2004, Floreano *et al.*, [15], imitating strategies observed in simple insects used a GA to tune up a neural network which visually recognizes edges, corners and height. This active vision system acts as an artificial retina, moving and focusing on important features. It was tested with the open source simulator Car-World (http://carworld.sourceforge.net) and the best evolved individuals performed equal to or better than well-trained human drivers tested on the same circuits.

In 2005, Sun *et al.* [16], used a GA to optimize the parameters of a set of Gabor filters in the context of vehicle detection from images. They tested the proposed framework on real data with success and improved the performance of on-road vehicle detection.

In the same year another interesting approach was proposed for automated evolutionary design of driving agents, [17] [18] (Figure 2). This work showed how GAs can help in the task of designing an agent able to remotely operate a scale racing car. The agent perceives the environment from a camera mounted overhead (position, orientation, velocity, approach angle, distance to the apex and outside/inside slow down zone). With these perceptions the agent sends commands to the remote control car (forward, neutral, reverse, left, straight or right). The comparative analysis established that on long runs the agent's operated car was 5% slower than the human operated one.

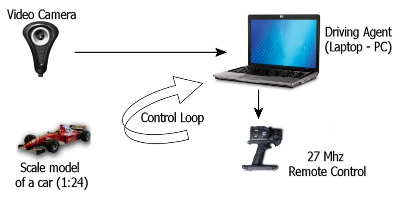

Fig. 2. Scheme of the system configuration for the 1:24 scaled model used in [18].

Working with the evolving weights of a neural network, Julian Togelius *et al.* compared, with their own simulator, that simulated cars with evolved neural network controllers (in first-person and third-person) [19], [20]. They extended their work to a more complex case of two cars competing against each other in the same track at the same time, [21], using evolutionary strategies to solve the problem of the co-evolution. Finally, an interesting study which compares neuroevolution and genetic programming in the same environment can be found in [22].

2) Vehicle parameter optimization: the study of the suspension system has been an interesting topic for researchers because it contributes to the car's handling and braking for good active safety and driving pleasure, and keeps vehicle occupants comfortable and reasonably well isolated from road noise, bumps, and vibrations.

In 1996 Cambiaghi *et al.* presented the results of a study conducted on the vertical dynamics of a two wheel car model [23]. They applied a GA to damper's optimization with their own simulator. The results showed a significant improvement in performance and comfort level compared to the traditional damper behavior. Unfortunately no experimental results with real vehicles were available.

Another relevant study about GAs applied to suspension systems is [24]. This work uses parallel GAs to optimize three suspension key parameters: longitudinal center of gravity, roll stiffness distribution and aerodynamics downforce distribution. The GA tests the parameters in a simulator designed by the authors and promotes the fastest individuals. The study helped in finding which values of the parameters were the most adequate and what the relationship between them was.

Finally, a recent work, [25], showed how to optimize 66 variables from a Formula one racecar with a GA. The parameters affect to the suspension system, the engine revolutions limits and gear ratios, the aerodynamics, and the tyre and brake modelling. These variables were tested in Electronic Arts' Formula One Challenge '99-'02. This is a commercial racing simulator released in 2003 with an advanced real time physics engine. The results of this work were of interest, the best car setup found involved performance improvements (faster lap times) compared to the default system setup or expert players' settings.

II. Objectives

The objective of the developed controller is to drive as good as possible in unknown circuits alone or in the presence of other drivers. To this end we decided to participate in the racing car competition held in the IEEE World Congress on Computational Intelligence (WCCI 2008).

A. TORCS Simulator

The software used (TORCS: The Open Racing Car Simulator, source: http://torcs.sourceforge.net/) is today one of the most popular simulators. This simulator is written in C++ and can be downloaded under GPL license from its web page. The source code and the executable files are available for Windows and Linux (for MacOS, only binaries are accessible), which allowed the competitors more flexibility to develop the controllers. The advantages of using this racing simulator are the following:

- TORCS has a high level of realism in graphics and physics.
- It provides a large quantity of vehicles, tracks and controllers.
- There is a large community of users and developers, which helps the program to be updated and be kept to a small number of errors.
- It is easy to develop a controller and to integrate it within the simulator.

However, one of the main problems of this simulator is a memory leak each time the race is restarted. This is an important disadvantage when the learning algorithm requires a large number of iterations or evaluations, where each one of them needs the race to be restarted. For instance, when a GA needs to evaluate a whole population of individuals, it will restart the race for each one of them, causing an increment of this memory leak and, finally, inducing the program to crash.

For this competition, the simulator has been adapted to offer a common interface to all competitors. A server bot is executed by the simulator, while each participant tests his controller as clients. The communication among them is designed using UDP packages, so each competitor can develop his controller by choosing his favourite programming language, with the sole condition of obeying the communication protocol.

B. Controllers, Sensors and Effectors

The interface provided by the server bot offers the client a total of 17 sensors and 5 effectors. The sensors include different kinds of information such as the angle of the car, the position, speed, location of opponents, etc. The effectors used to drive the vehicle are the steering wheel value, both pedals (throttle and brake) and gearing change. A full description of all the sensors provided can be seen in the bases of the competition (documentation, sources and examples are available at http://cig.dei.polimi.it/?page_id=5).

Five controllers were presented to the competition and three more were provided by the organizers. The aim of the programmed controllers is to make a comparison against the results obtained by the automatically developed ones.

C. Controller evaluation

The evaluation of the controllers results from their performance on three different test circuits made by the organizers. The tests are done in different tracks which are previously unknown by the competitors and divide the tournament into two stages. At the first stage, every controller is thrown alone in each circuit, evaluating the distance raced in $10,000$ game tics. To obtain trustable results, ten executions are performed on each track, considering the mean of all of them for point assessing. In the second stage, all controllers (except the three programmed by the organizers) are tested together, again ten times, measuring the arrival order to evaluate the drivers.

In both cases, the points assigned to each controller follows the Formula One World Championship pointscoring system (10 points for the first, 8 for the second, 6 for the third, and so on, up to one point for the eighth).

III. Controller Design

The design of the controller is explained in this section, including a description of problems found and decisions taken through the development.

A. Input Data

Among all the available sensors, only four of them have been used to create the rule system. These input data were discretized in order to obtain a set of rules used by the vehicle to drive in any circuit. The sensors used are the following:

- Angle (Figure 3): angle between the car direction and the direction of the track axis. The target range is [0,4], where 0 means small angles and higher values represent larger ones.

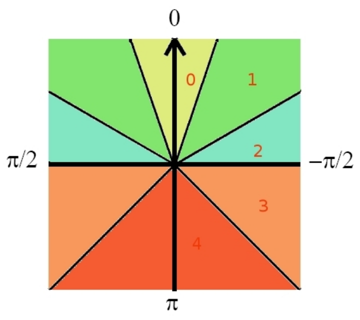

Fig. 3. Vehicle angle with track axis discretization.

- TrackPos (Figure 4): distance between the car and the track axis. This continuous value has been discretized in a discrete range [0,1], where 0 means centered in the track and 1 represents the car near the edges.

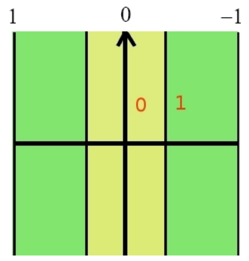

Fig. 4. Vehicle track position discretization.

An important feature of the design is the usage of symmetry for the first two sensors. This concept works by using the absolute value of the sensor to match to the proper discretized value. In other words, a value of $-\alpha$ and α from any of those sensors will mean the same discretized value. The objective of this approach is two-fold: to reduce the search space and to avoid the algorithm learning how to face similar situations twice, taking into account that the difference is only the sign of the sensor value.

- SpeedX (Figure 5): speed of the car along the longitudinal axis of the car, discretized in a range [0,3], where 0 means lower speed than higher values.

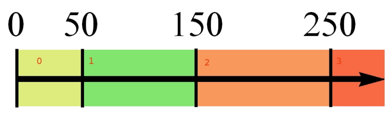

Fig. 5. Vehicle speed discretization.

- Track: the set of sensors that indicates the distance to the track edges. Only three of them have been used (the front one and the immediate sensors on its right and left) and they have been discretized to a unique range [0,2]. In this case, 0 means that the track edge has been detected between 20 and 100 meters, 1 means that the track edge is up to 20 meters and 2 that no track edge is seen.

B. Effectors

The effectors of the controller have been designed as follows:

- Throttle and brake: both pedals have been codified in a common output, in order to avoid non-sense values. For instance, a full gas value in both pedals at the same time would produce uncertain (and useless) outcomes. Hence, a unique value ('a') is applied and both gas pressures are extracted from it using the formula depicted in the Figure 6.

$$\text{throttle:} \begin{cases} 0 & \text{a<0.5} \\ \text{(a-0.5)·2} & \text{otherwise} \end{cases}$$

$$\text{brakes:} \begin{cases} \text{1-(a·2)} & \text{a<0.5} \\ 0 & \text{otherwise} \end{cases}$$

Fig. 6. Formula for pedals values.

- Steer: the steering output is codified as a real number, taking the same values as the effector, from -1 to 1, but discretized with a precision of 0.1. The reason for this discretization is to reduce the number of possibilities, considering only the ones with most relevance: notice that a difference of ± 0.001, for instance, in steering does not become in an important change on how the car steers.
- Gear: gear changes did not take part in the learning process, and have been defined as follows: if the revolutions per minute of the engine are higher than $6,000$, then the gear value is increased by one. However, if this value is lower than $3,000$, the current gear is decreased. Otherwise, the gear does not change.

C. Rules

The discretization applied over the input data allows us to create a set of 120 rules (Figure 7), where the conditional part is composed of the above four sensors used and the actions are formed by acceleration, braking and steering, which are codified as seen previously. This set of initial rules are the basis of the base individual.

D. Base individual

One of the biggest problems found when evolutionary techniques are applied to obtain autonomous driving is that traditional random initialization of individuals does not

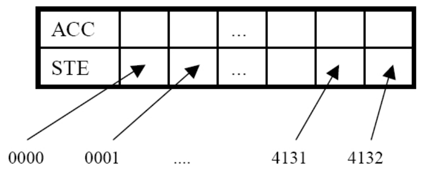

Fig. 7. Set of rules scheme.

work properly because it is almost impossible to obtain a configuration that drives the vehicle correctly by chance. This is the reason for getting a base individual capable of finishing at least one lap in the circuit, and then evolve it to obtain better results.

The algorithm used to obtain this base individual is the generation of a subset of rules that allows the vehicle to end a lap, minimizing the angle of the car with the track axis, no matter the speed of the vehicle. Each one of these rules is created by testing how each allowed combination of acceleration and steering behaves when the left part of the rule (condition) is triggered. The result of this procedure is a "novice" controller who drives at a low speed and always centered on the track.

E. Evolutionary Rule System

Once the base individual is obtained, an evolved individual is generated from it, taking all the rules used in the previous process. Obviously, not every one of the set of 120 rules is used, so the final individual will only use a subset of them (results show that this number is usually between 10 and 20), depending on the circuit used to obtain the base individual. Therefore, the individual is composed of a set of rules, each one of them formed by the sensors condition and the effectors action, that need to be evolved to obtain the controller. An scheme of this individual is depicted in Figure 8.

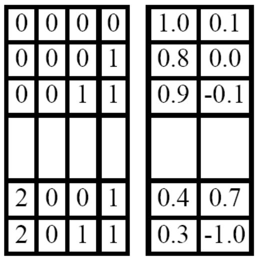

Fig. 8. Base individual scheme.

To evolve this individual, a circuit is chosen (complex enough to provide multiple and different driving scenarios) and the following algorithm is executed:

The evaluation of the individual is performed by recording the lap time and damage suffered when it is executed in a circuit for three laps. The fitness assigned to the set of rules is obtained through a lineal combination of both values, with weights of 0.4 and 0.6 respectively. This adjustment is made in order to avoid the overfitting to the training circuit. The objective here is not to obtain the best lap time on the training track, but to obtain a vehicle capable of taking the turns correctly. On one hand, if the only factor to take into account in the evaluation of the individual were the lap times, then the vehicle would learn how to handle along the bends of this specific circuit, as did in [26], and it would be useless for other tracks. On the other hand, if the car suffers a lot of damage, it will end up stopped somewhere in the circuit (and that is negative if we wish to drive many meters, as requested in stage one of the competition). Thus, this parameter was then included in the fitness evaluation.

In this rule system, we can not decide when a rule is better than the other because the behaviour of the individual depends on the whole set of rules used. Because of this, a selection operator has been designed as a random pick-up from the pool of rules, taking two of them to apply crossover. This operator has been implemented as an uniform crossover, using a probability of 50% to choose a gene from each parent rule. Finally, a mutation is performed over the new rule, applying an addition of ± 1 unit to the left part and ± 0.3 to the effectors of the rule (always obeying the limits of the genes and their codification precision, as was specified in previous sections).

The next step in this algorithm consists of the search of the rule in the individual with identical conditional part to the new one. This rule is extracted from the pool and the new one replaces it. Notice that the impact of this substitution is not too high because of the similarity protocol. After that, the new set of rules is evaluated, and the fitness is compared with the one calculated before inserting the new rule until a stop criterion is met. If the new set obtains a better fitness, the new rule stays in the individual. Otherwise, the substituted rule is retrieved and the new one is eliminated. In any case, a new step of the algorithm is executed until a stop criterion is reached.

Each one of the steps of this algorithm is shown in the scheme of Figure 9.

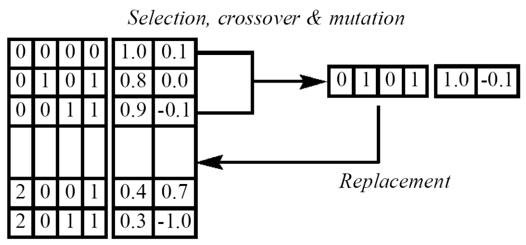

Fig. 9. Algorithm step.

Results shows that this algorithm is effective since it reduces the base individual's lap times in few generations, and keeps the damage of the car almost inexistent. However, more tests need to be done in order to extract some conclusions.

IV. Results

The controller developed for the competition (called DIEGO) was tested in three circuits and it obtained acceptable results in the first two (Figures 10 and 11).

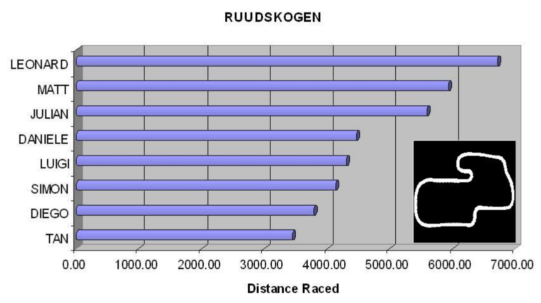

Fig. 10. Ruudskogen circuit results at Stage 1.

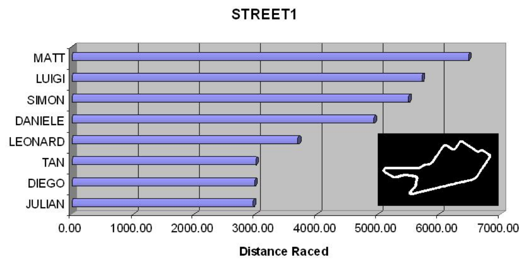

Fig. 11. Street1 circuit results at Stage 1.

After studying the results we concluded that the usage of symmetry produced a side effect that was not expected: the car drives in a smooth zig-zag trajectory centered on the track. This is because a small steering value can center the vehicle on the track, but not necessarily drive it parallel to the track axis, because symmetry changes the sign of the steering value when the track axis is crossed. However, the controller behaved in a reasonable way, only with just the exception of some specific circuits: the oval ones. These circuits, similar to Nascar tracks, have banked bends which make the zig-zag movement completely uncontrollable.

This problem affected dramatically to the behaviour of this driver in the third circuit of the test bed, as can be seen in the results shown in Figure 12.

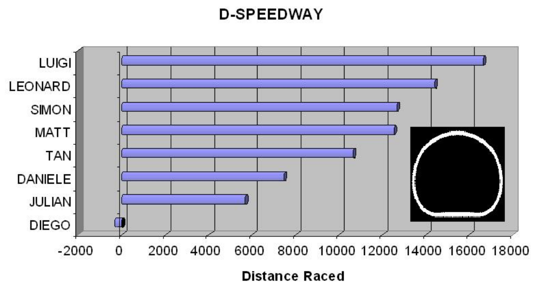

Fig. 12. D-Speedway circuit results at Stage 1.

In the second stage, the races performed again over these circuits gave the last set of points assigned to the drivers. The final results can be seen in Table I. Note that the manually programmed controllers are not included (LUIGI, DANIELE and JULIAN)

TABLE I
FINAL RESULTS.

	RUUDSKOGEN	STREET1	D-SPEEDWAY	TOTAL
MATT	10	10	6	26
LEONARD	4	8	10	22
SIMON	6	6	8	20
TAN	5	5	5	15
DIEGO	5.5	4.5	4	14

The results of the races are very similar to the results obtained in the first two circuits on stage 1, far from the best controllers, but with reasonable positions. However, the results obtained in the third circuit on stage 1 weighted the final score in such a way that finishing in higher positions became impossible.

As seen before, the main difference between the *D-Speedway* circuit and the others in the bed test was the presence of banked bends. For that reason, another attempt, which considered an oval circuit (*B-Speedway*) with these kind of banked bends for the training process, was made after the competition so the vehicle could adapt better to these conditions.

After these modifications, the experiment results for the three tracks of the competition yielded: $5,297$ meters on *Ruudskogen*, $4,499$ meters on *Street1* and $5,670$ meters on *D-Speedway*. These results cannot be directly compared with those obtained by all other competitors due to hardware factors, technical features of the computer used for the purpose of this work and the one used for the competition are different, thus, some variations are expected. However, these results showed that the driving of the vehicle improved using this kind of training circuits. After obtaining these results we expect great improvements in the future controllers when training them with more circuits.

V. Future Work

Considering the limited time available to prepare the controller for the competition, we can evaluate our work as positive, since we only failed in one of the test circuits. However, the main aim now is to continue working on this driver in order to obtain a very good controller.

Competition organizers decided to propose a new edition of this tournament for the IEEE Symposium on Computational Intelligence and Games (CIG'2008), so the main objective now is to improve this controller in order to obtain better results for this new call. The next steps to follow are:

- Discretization: the input data discretization was performed by hand, without using any methodology that help us to determine the target ranges. The usage of a technique of clustering (for instance, applying k-means over human driving in the simulator) could determine more representative ranges.
- Symmetry: as has been previously said, symmetry caused a big problem in some circuits, despite the fact that it has decreased the search space and helped the evolutionary algorithm to obtain better results. The next

controller will introduce a system without symmetry, or using a limited symmetry, to avoid the zig-zag movement of the vehicle. If we finally get better discretization ranges, as stated in the last point, the impact on the search space would not be a problem for the final design.

- Evaluation in more than one track: TORCS simulator does not easily allow restarting the race in a circuit different than the one used for starting the execution. This causes the learning to be performed in only one circuit, which can produce overfitting on the training track, despite the attempts to choose a complex circuit to force the vehicle to face a set of different scenarios (such as bends, track widths, circuit lengths, etc.). The idea is to use more than one circuit in the learning process collecting more efficient rules for the controller in all circumstances.
- Opponent analysis: an interesting point observed in race competition videos is that most of the delivered controllers can not avoid the opponents when they are about to overtake the other cars. Instead of that, the vehicles hit them in the back without evading them. For the next competition, an analysis of these opponents will be made in order to obtain a competitive advantage in the race phase.

ACKNOWLEDGMENTS

This work was supported in part by the Spanish MCyT project OPLINK, under grant no. TIN2005-08818-C04-02.

REFERENCES

[1] R.E. Fenton. *IEEE Transactions on Vehicular Technology*, 19(1):153–161, 1970.

[2] J. Bernard, J. Gruening, and K. Hoffmeister. Evaluation of vehicle/driver performance using genetic algorithms. *SAE International Congress and Exposition*, (980227), 1998.

[3] A. Niehaus and R. F. Stengel. Probability-based decision making for automated highway driving. *Vehicle Navigation and Information Systems Conference, 1991*, 2:1125–1136, Oct. 1991.

[4] G. Siegle, J. Geisler, F. Laubenstein, H. Nagel, and G. Struck. Autonomous driving on a road network. *Intelligent Vehicles '92 Symposium., Proceedings of the*, pages 403–408, Jun-1 Jul 1992.

[5] R. Sukthankar, D. Pomerleau, and C. Thorpe. Shiva: Simulated highways for intelligent vehicle algorithms. In *In Proceedings of IEEE Intelligent Vehicles*, pages 332–337, 1995.

[6] D. Pomerleau and T. Jochem. Rapidly adapting machine vision for automated vehicle steering. *IEEE Expert*, 11(2):19–27, Apr 1996.

[7] R. Sukthankar, J. Hancock, S. Baluja, D. Pomerleau, and C. Thorpe. Abstract adaptive intelligent vehicle modules for tactical driving. In *In Proceedings of AAAI-1996 Workshop on Intelligent Adaptive Agents.*

[8] C. Thorpe, T. Jochem, and D. Pomerleau. Automated highway and the free agent demonstration. In *In Proceedings of 1997 IEEE Conf. on Intelligent Transportation Systems*, pages 496–501, 1997.

[9] M. Bertozzi, A. Broggi, G. Conte, and R. Fascioli. The experience of the argo autonomous vehicle. In *in Procs. SPIE'98 - Aerosense Conf*, volume 3364, pages 218–229, 1998.

[10] J. M. Collado, C. Hilario, A. de la Escalera, and J. M. Armingol. Self-calibration of an on-board stereo-vision system for driver assistance systems. *Intelligent Vehicles Symposium, 2006 IEEE*, pages 156–162, June 2006.

[11] Q. Chen, U. Ozguner, and K. Redmill. Ohio state university at the 2004 darpa grand challenge: developing a completely autonomous vehicle. *Intelligent Systems, IEEE*, 19(5):8–11, Sept.-Oct. 2004.

[12] S. B. and R. Caruana. Removing the genetics from the standard genetic algorithm. In *The Int. Conf. on Machine Learning 1995*, pages 38–46, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[13] S. Baluja, R. Sukthankar, and J. Hancock. Prototyping intelligent vehicle modules using evolutionary algorithms. pages 241 – 257, 1998.

[14] L. D. Pyeatt, A. E. Howe, and C. W. Anderson. Learning coordinated behaviors for control of a simulated robot. Technical report, Computer Science Dept, Colorado State Univ., Ft. Collins, CO 80523, 1996.

[15] D. Floreano, T. Kato, D. Marocco, and E. Sauser. Coevolution of active vision and feature selection. *Biological Cybernetics*, 2004.

[16] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, (6):125–137, 2005.

[17] I. Tanev, M. Joachimczak, H. Hemmi, and K. Shimohara. Evolution of the driving styles of anticipatory agent remotely operating a scaled model of racing car. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 2:1891–1898 Vol. 2, Sept. 2005.

[18] I. Tanev, M. Joachimczak, and K. Shimohara. Evolution of driving agent, remotely operating a scale model of a car with obstacle avoidance capabilities. In Mike Cattolico, editor, *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1785–1792, New York, NY, USA, 2006. ACM.

[19] J. Togelius and S. M. Lucas. Evolving controllers for simulated car racing. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 2:1906–1913 Vol. 2, Sept. 2005.

[20] J. Togelius and S.M. Lucas. Evolving robust and specialized car racing skills. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1187–1194, 2006.

[21] J. Togelius and S. M. Lucas. Arms races and car races. In *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Procedings*, volume 4193 of *Lecture Notes in Computer Science*, pages 613–622, 2006.

[22] A. Agapitos, J. Togelius, and S. M. Lucas. Evolving controllers for simulated car racing using object oriented genetic programming. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1543–1550. ACM Press, 2007.

[23] D. Cambiaghi, M. Gadola, L. Manzo, and D. Vetturi. Semi-active strategies for racing car suspension control. *SAE International Congress and Exposition*, (962553), 1996.

[24] E. M. Kasprzak, K. Hacker, and K. Lewis. Exploring the design tradeoffs and computational savings of executing vehicle simulations in a parallel computing environment. In *ASME Design Automation Conference, DETC00/DAC-14243*, 2000.

[25] K. Wloch and P. J. Bentley. Optimising the performance of a formula one car using a genetic algorithm. In *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 702–711, 2004.

[26] Y. Sáez, D. Perez, O. Sanjuán, and P. Isasi. Driving cars by means of genetic algorithms. In *PPSN*, pages 1101–1110, 2008.