The Epic Adaptive Car Pilot

Charles Guse and **Dana Vrajitoru**
Computer and Information Sciences Department
Indiana University South Bend
South Bend, IN 46617

Abstract

In this paper we present an adaptive autonomous car pilot developed for the TORCS competition environment. This software provides a multi-track competition system where the user can control a car on several different circuits and compete against other pre-defined pilots. Our pilot is composed of several control units based on the input available within the TORCS system. The parameters of the systems were learned by a hill-climbing method and the pilot can adapt some of them dynamically to a new track. The car was submitted to the GECCO 2009 competition (Raidl 2009).

1. Introduction

Automatic pilots have been a standard in aviation for a good number of years (Atkins et al. 1998), and the interest is slowly but surely being extended to cars and other vehicles (Al-Shihabi and Mourant 2003), (Chaperot and Fyfe 2006), (Gavrilets et al. 2001). With the amount of traffic present on the road, assistive driving systems are becoming more common and popular, and so the need to develop autonomous or semi-autonomous pilots is present in the industry.

In this paper we present an adaptive car pilot developed within the TORCS (The Open Racing Car Simulator) environment (Wymann and Espie) (Lazaric et al. 2007). The problem of developing autonomous car drivers is challenging and intensive. Being able to test your system in a competition setting, where it can be compared with similar systems from around the world, makes it even more interesting. Using the TORCS environment for the project was a good way to develop our driver in a highly realistic simulation both in terms of graphics and of physics. This has helped us write a driver that is more potentially useful in the real world. The program makes it easy for new users to engage and focus on the high level details instead of the physics implementation.

The TORCS system provides a simulation environment for realistic car driving and competing, with appealing graphics and complex physics. The program provides a server for the race and the user can write a module that can be plugged in as a client. Our pilot was submitted to the Genetic and Evolutionary Computation Conference in 2009 (Raidl 2009). The TORCS system has been used for several competitions already (Loiacono et al. 2008) and this paper presents our first experience with it. The best drivers have reported the use of evolutionary strategies using covariance matrix adaptation to improve the system (Butz and Lönneker 2009), or fuzzy systems (Onieva et al. 2009), or the use of neural networks to learn by imitating a human driver (Muñoz 2009).

The model developed for the car race bears some similarity with the motorcycle pilot driver presented in (Vrajitoru and Mehler 2005). Similar ideas are used in the decomposition of the pilot in control units, but these components also come naturally from the way a real vehicle is driven. Some of the decisions taken on the road follow common strategies, but the available perception data are quite different between the two systems and the control outputs are also substantially different.

Our model consists in decomposing the car control into several units, mostly following the control outputs specified within the TORCS system. Thus, our pilot starts by finding a suitable target direction, based on which it will determine the target speed, which in turn determines the gear, acceleration, and brakes. The driving decisions are based on concerns about maximizing the available free distance ahead, keeping the vehicle safely inside the road, and anticipating sharp turns further down the road. The pilot presents a learning and adaptive component both to choose appropriate values for all the parameters it uses, and to adapt to a new track when the need occurs.

2. The TORCS Competition Environment

TORCS (Wymann and Espie) (Lazaric et al. 2007) is a multi-platform open source racing car simulation environment created by a multi-national team. It has an active community of developers and users and competitions are organized yearly to take advantage of it as part of several international conferences.

The software provides a variety of car models and tracks for the race and allows up to 50 opponents to compete against each other. The race takes place in a realistic 3D environment with sophisticated dynamics where the terrain configuration and the car specifications will greatly influence the physical behavior of the vehicle on the road. The conditions become even more complex as the cars interact with each other and can collide and damage one another. Figure 1 shows a screen capture of this environment.

Figure 1: The TORCS Graphic Environment

The system is organized in a client-server mode. The program provides a server that keep track of all the vehicles involved in the race. The user can write a C++ or Java API module that can be compiled to run as a client to the TORCS server. Figure 2 shows the outline of the system, where UDP is a socket-based communication API.

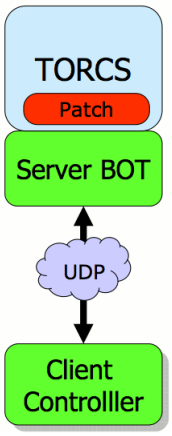

Figure 2: The TORCS Interface

Communication.

The client will receive a set of perception indicators as *input* from the server in real time. These consist in the following measures:

- the distance to the border of the road or to the closest obstacle in a direction going from -90^o to $+90^o$ from the direction of movement of the vehicle, in a fan covering the area in 10^o increments; the visibility for this measure is limited to 100m;
- the distance to the closest opponents in a 360^o area, also in 10^o increments and limited to a radius of 100m;
- the current speed, gear, angle with the centerline of the road, RPM, fuel, damage, distance covered in the race, position on the track, etc.

Figure 3 shows an example of the opponent sensors for a car with another vehicle close by.

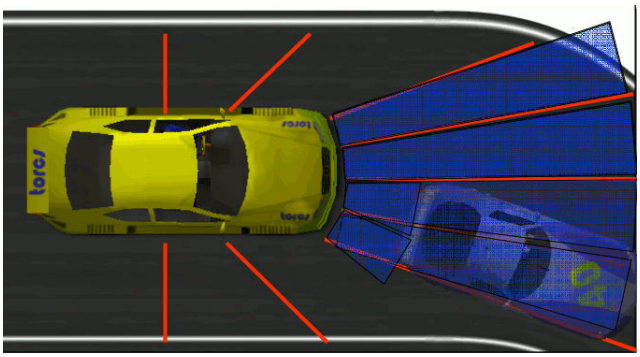

Figure 3: Opponent sensor information

The control *output* from the client must consist in the following *effectors*:

- *steering wheel* [-1,+1] for a change in direction: -1 corresponds to -45^o while +1 to 45^o; this simulates the technical specifications of a real car that couldn't turn by an arbitrary angle on the spot,
- *gas pedal* [0, +1] for accelerating; just as for a real car, a lack of gas input will result in loss of speed;
- *brake pedal* [0,+1] for slowing down,
- *gearbox* {-1,0,1,2,3,4,5,6} for a change of gear.

3. Epic Car Pilot Model

In this section we present the details of the car pilot that we developed for this competition.

The control of the car is decomposed in several control units:

- the target speed; this is the speed we will attempt to achieve and is considered ideal for the current road conditions;
- the gear, a simple function of the target speed,
- the target direction, in reference to the current direction,
- speed adjustment for sharp turn; this last component anticipates a sharp turn further down the road and not in the immediate vicinity of the car.

The general algorithm consists in the following steps:

- calculate the target direction and speed
- determine the correct gear
- calculate the target angle based on the target direction
- calculate the acceleration and brakes based on the target angle and speed

Below we will describe each of the control units more in detail.

3.1 Gear

We have used a simple gear changing mechanism where if the RPM value goes up to about 7000, then we move to the next gear up, and if the RPM descends to about 3000, then we move to the next gear down.

3.2 Target Direction

This component of the car pilot decides the direction of movement of the car. More specifically, the function responsible for steering will return a target angle as a result. In this case, 0 means that no change is necessary to the current direction, while any other value will cause the car to steer.

To compute the target direction we start by deciding if the car can continue to move in the current direction. The conditions for not altering the direction are:

- if the current direction of the car is close enough to the direction of the road centerline,
- if there is enough free distance straight ahead in the car's direction of movement
- if the car is safely inside the track, meaning at a given percentage (for example, 90%) of distance to the borders of the road.

If the three conditions above are not met, then we need to decide on a new direction of movement. For this, we start from the direction of the road centerline, and scan by increments of 10 degrees in the direction in which the available free distance ahead of the car increases or stays constant, until we find an angle at which this distance starts to decrease, or until we reach the maximal allowed turn angle of 45^o from the current direction of movement. Figure 4 illustrates this scanning procedure to look for a good direction of movement:

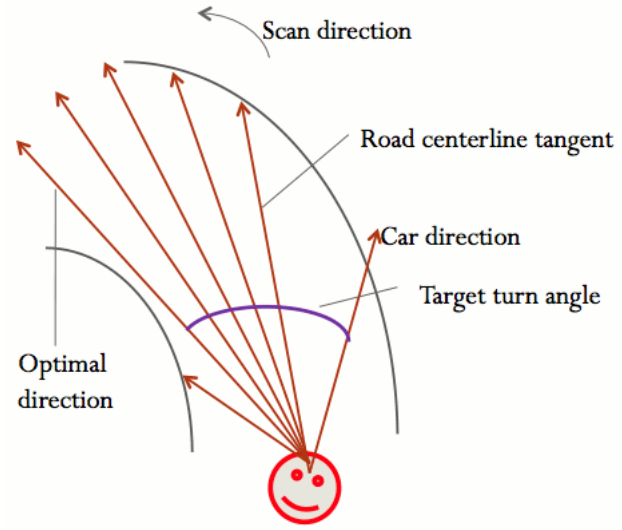

Figure 4: Determining the target direction of movement

The second factor in deciding the new direction of movement is the closeness to the border of the road. If the car is too close to the border of the road or if it is outside the road altogether, we modify the steering angle to bring the vehicle back inside the road. The threshold for the car being safely inside the road was established experimentally as 90% of the distance between the center of the road and the border. We needed to adjust this component such that when the car is turning in one direction and following the inside curve, it does not get constantly pushed away from it, as shown in Figure 5.

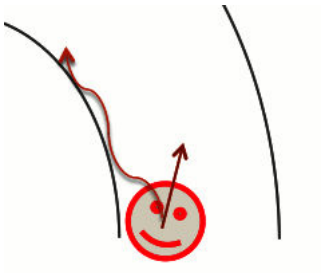

Figure 5: Constant wiggle around a curve

3.3 Target Speed

The target speed is computed after the steering angle has been decided at the previous step, because the speed at which the car can safely ride depends on how straight it is going.

The safe conditions for the speed are considered to be:

- if the car is going almost straight, meaning if the target angle for the change in direction is smaller than a given threshold;
- if the free distance in front of the car in the new direction given by the target angle is large enough;
- if no sharp turn is anticipated to follow up on the road soon.

If the three conditions above are met, then the acceleration is set to its maximal value. We call this situation *pedal to the metal*.

In any other case, we start with a large value for the target speed, which is first scaled by the sine of the target angle for the change in direction and with the available free distance in the aimed direction.

3.4 Sharp Turn Factor

An early difficulty we observed was that the car skidded when we tried to turn by a large amount at a high speed. To illustrate this particular condition, we have taken some measurements of the difficulty of a turn with respect to the speed of the vehicle when engaging it. The track we used, illustrated in Figure 6, was named Aalborg. We've run the car by hand over four of the angles of various difficulty and at various incoming speed. For each angle we've registered the speed range within which the turn could be taken safely, or could be achieved with some amount of trajectory adjustment required in addition to turning, and at which a collision combined with skidding would ensue. Table 1 shows these measurements. The four turns we've chosen for the experiment are marked on the images by circles. One of them is marked by two angles in the table because it is composed of two really close turns of different angles.

If the vehicle tomes toward at a high speed, it may not be able to slow down fast enough to take the turn safely. We concluded that a sharp turn needed to be detected ahead of

Figure 6: Aalborg track used to measure the safe speed for entering a turn

Table 1: Speed range (km/h) while entering a turn and driving safety

Angle	Safe	Correction	Collision
84^o	0-44	45-55	> 55
$106^o + 75^o$	0-54	55-70	> 70
106^o	0-74	75-105	> 105
112^o	0-79	80-110	> 110

time, and not just when it comes within turning range. Thus, we computed the *sharp turn factor* as 1 over 1+ the minimal absolute difference between the distance ahead at adjacent angles in 10 degrees increments, as explained below. This factor is used to further scale the speed after it is computed as detailed in the previous subsection.

The factor is computed to account for the sharpest turn in the road detected ahead. For this purpose we scan up to 20 degrees left and right of the aimed direction and we look at the difference between the free distances ahead in adjacent directions that differ by 10 degrees. A very similar amount of free distance indicates a sharp turn in the road, because the new direction of the border of the road in that case is at a large angle with the current direction of movement. This is a situation that requires the vehicle to slow down ahead of time because it can happen that at a high speed, the car will not be able to steer fast enough to make the turn. A larger difference between these distances indicates that the road continues in a direction that is close enough to the current direction of movement, so it is safe to go at a higher speed. Figure 7 illustrates these two situations.

3.5 Learning

Each control unit of the car is controlled by several parameters that can be adapted and learned with any given machine learning method. As this was our first experience with the

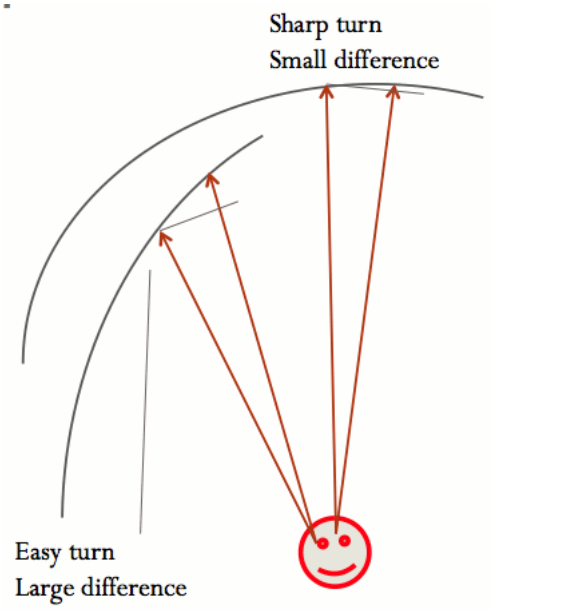

Figure 7: Detection of a sharp turn

competition, we used a simple hill-climbing technique to adjust these parameters. As future research we would like to employ simulated annealing or genetic algorithms to learn these parameters.

The pilot also has a dynamic adaptation mechanism that it can use to learn its behavior on a new track. Each race on a given track in the competition settings consist of several circuits completed on the same track. The pilot will first complete a circuit on the new track and examine the results. If no damage has been registered during this first trial, the parameters determining the maximal speed in each situation and the caution to be taken during turns on the road are incremented to make the car go faster. Otherwise every time the car gets out of the track or registers damage without another car being close by, the pilot will adapt the same parameters to make its behavior safer.

4. Independent Testing

The TORCS environment provides some numerical measures that are reported after each competition is completed. As a benchmark for future research and to see how well our system is doing, Table 2 shows some of these reports on several races of increasing difficulty. We have tested our system in a competition involving two other pre-program drivers available in TORCS: a very fast one called Damned 1 and a safer one called Inferno 2. For this first set of experiments, the adaptation/learning mechanism was not active.

The track called E-Track 3 is the one that we've trained our system on. The tracks called Alpine 1, Dirt 3, and Eroad are the ones used in the GECCO competition. The track called Street 1 was particularly difficult because two very sharp turns were bordered by an area of sand, and if the car skidded onto it, it would take a long time to navigate back to the road.

Table 2: Driver ranking and statistics on several tracks without adaptation

Track	Laps	Best Time (s)	Top Speed	Damage
E-Track 3				
1. Damned 1	10	101.97	231	0
2. Epic	10	112.36	254	74
3. Inferno 2	9	122.45	194	0
Alpine 1				
1. Damned 1	10	151.22	131	783
2. Inferno 2	9	200.74	180	1083
3. Epic	8	186.12	193	5483
Eroad				
1. Damned 1	10	75.93	219	0
2. Inferno 2	9	89.86	192	0
3. Epic	9	89.58	213	944
Dirt 3				
1. Damned 1	10	65.01	185	1002
2. Inferno 2	9	78.72	152	694
3. Epic	8	69.12	156	2096
Street 1				
1. Damned 1	10	92.01	230	1
2. Inferno 2	9	107.45	198	0
3. Epic	1	457.38	218	162

Table 3: Driver ranking and statistics on several tracks with adaptation/learning

Track	Laps	Best Time (s)	Top Speed	Damage
E-Track 3				
1. Damned 1	10	101.89	231	0
2. Epic	9	114.02	242	676
3. Inferno 2	9	122.51	194	0
Alpine 1				
1. Damned 1	10	149.79	216	6
2. Inferno 2	8	186.4	175	3716
3. Epic	7	196.48	190	3828
Eroad				
1. Damned 1	10	75.5	220	0
2. Inferno 2	9	90.01	191	411
3. Epic	8	100.5	205	808
Dirt 3				
1. Damned 1	10	64.61	182	1139
2. Inferno 2	9	78.57	153	686
3. Epic	5	130.25	160	596
Street 1				
1. Damned 1	10	91.53	230	0
2. Inferno 2	9	107.56	193	1048
3. Epic	6	126.74	201	2330

Our driver outranked one of the two drivers we raced against on the track we've used for learning and came close to the performance of the best driver on this track. Its performance was not as good on the other tracks, which means that for the next development we need to focus on several tracks at the same time. From a visual observation of the race, most of the recorded damage comes from collision with the other vehicles, so a more sophisticated avoidance mechanism could improve our driver's performance.

Table 3 shows the statistics of similar experiments performed with the learning mechanism that adapts the speed function to the road conditions activated. By comparing the results of the Epic driver on all the tracks, we can see that the amount of damage incurred by the car has significantly decreased in all the cases. In particular, on the difficult track Street 1, a more conservative speed behavior has allowed the car not to skid on the sand area too often. Thus, the car was able to complete 5 laps before the winner completed 10, as opposed to only 2 without adaptation.

These results are encouraging for continuing to improve the adaptation mechanism for future work. However, this adaptation has not allowed the car to outperformed the best provided pilot on any of the tracks. Furthermore, in many cases, the driver was slowed down by this mechanism and the number of completed lapses decreases in some cases. Based on these observations, we decided to submit our driver to the GECCO competition without adaptation.

5. Competition Results

We submitted the car pilot to the ACM Genetic and Evolutionary Computation Conference 2009 for their TORCS competition. This was our very first experience with such a competition and many of our competitors were veterans pilot developers.

The competition consisted of two stages. Each stage was composed of 3 tracks and a maximum of 10 circuits to complete on each track. In all the cases the cars would run together until one of them completed the 10 circuits of the track which were the goal. When that happened the race would be over and the score recorded for each car was the number of completed circuits up to that point. The total score over the 3 races is a sum of the scores on each track.

In the first stage all the pilots that were submitted to the competition raced against each other, and the goal was to select the top 8. Figure 8 shows the results of this first stage. Our pilot, called Epic, qualified to the second race in the 7th place.

Rank	Competitor	Dirt 3	Alpine 1	Eroad	Total
1	COBOSTAR	8	10	10	28
2	Onieva&Pelta	10	8	8	26
3	Luigi (Champ2008)	4	4	5	13
4	Perez&Saez	3	6	2	11
5	Mr. Racer	5	5	NQ	10
6	DRT	1	3	6	10
7	Epic	6	0	1	7
8	Simplicity	2	1	3	6
9	Jorge Fuentes	0	0	4	4
10	RedJava	0	2	0	2
11	Witold	0	0	0	0

Figure 8: Results after the first stage of the competition

The second stage was identical to the first one except that only the top 8 pilots participated. Figure 9 shows the final

results after this stage. Our pilot, Epic, was placed this time on the 4th place.

Competitor	Alpine 1	E-Road	Dirt 3	Total
Onieva&Pelta	10	10	12	32
Luigi (Champ2008)	7	8	8	23
COBOSTAR	5,5	7,5	3,5	16,5
Epic	5,5	3,5	4	13
Simplicity	3,5	4	5	12,5
DRT	3,5	5	2,5	11
Perez&Saez	4	3,5	3,5	11
Jorge Fuentes	3	2,5	3	8,5
Mr. Racer	0	0	0	0
Redjava	0	0	0	0
Witold	0	0	0	0

Figure 9: Results after the second stage of the competition

Our conjecture about these results and the difference between the stages, even though the tracks were the same, is that it is due to the fact that our pilot does not incorporate a sophisticated opponent maneuvering procedure. Thus, our pilot treats the opponents as simple obstacles or as if the border of the road was too close, and the avoidance procedure doesn't consider the fact that they are moving targets. During the first stage of the competition, more cars were present, and our car could more easily have been bounced off the track by one of these opponents. In the second stage, fewer opponents being present means that the car had less to deal with the other cars and thus our final results were better.

In the future we are planning to add a better learning system to improve the performance of our driver. The program depends on a large set of parameters whose values can be learned by any machine learning strategy. We plan to use simulated annealing as a first learning strategy for these parameters, and compare it with the use of hill climbing and genetic algorithms. The dynamically adaptive system that adjusts these parameters during the race can also be improved and extended.

6. Conclusions

In this paper we presented an autonomous car pilot capable of driving successfully over a variety of tracks. The pilot has a modest learning component and will be extended to grow in this direction.

The car pilot was submitted to the GECCO 2009 competition. It passed the first screening stage successfully and finished on the fourth place in the second stage.

The results are promising and we intend to continue working on our model and to participate again in similar competitions.

References

Atkins, E. M.; Miller, R. H.; VanPelt, T.; Shaw, K. D.; Ribbens, W. B.; Washabaugh, P. D.; and Bernstein, D. S. 1998. Solus: An autonomous aircraft for flight control and trajectory planning research. In *Proceedings of the American Control Conference (ACC)*, volume 2, 689–693.

Butz, M. V., and Lönneker, T. D. 2009. Optimized sensory-motor couplings plus strategy extensions for the TORCS car racing challenge. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 317–324.

Chaperot, B., and Fyfe, C. 2006. Improving articial intelligence in a motocross game. In *Proceeding of the IEEE Symposium on Computational Intelligence and Games*.

Gavrilets, V.; Frazzoli, E.; Mettler, B.; Piedmonte, M.; and Feron, E. 2001. Aggressive maneuvering of small helicopters: a human centered approach. *International Journal on Robotics Research*.

Lazaric, A.; Loiacono, D.; Prete, A.; Restelli, M.; and Lanzi, P. L. 2007. Learning driving tasks in TORCS using reinforcement learning. In *Machine Learning and Games (MALAGA) NIPS 2007 Workshop*.

Loiacono, D.; Togelius, J.; Lanzi, P. L.; Kinnaird-Heether, L.; Lucas, S. M.; Simmerson, M.; Perez, D.; Reynolds, R. G.; and Saez, Y. 2008. The WCCI 2008 simulated car racing competition. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.

Al-Shihabi, T., and Mourant, R. 2003. Toward more realistic behavior models for autonomous vehicles in driving simulators. *Transportation Research Record* (1843):41–49.

Muñoz, J. 2009. Controller for TORCS created by imitation. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 271–278.

Onieva, E.; Pelta, D. A.; Alonso, J.; Milans, V.; and Prez, J. 2009. A modular parametric architecture for the TORCS racing engine. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 256–262.

Raidl, G., ed. 2009. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Montreal, Canada: ACM SIGEVO.

Vrajitoru, D., and Mehler, R. 2005. Multi-agent autonomous pilot for single-track vehicles. In *Proceedings of the IASTED Conference on Modeling and Simulation*.

Wymann, B., and Espie, E. TORCS- the open racing car simulator. http://sourceforge.net/projects/torcs/.