Controller for TORCS created by imitation

Jorge Muñoz, German Gutierrez, Araceli Sanchis

Abstract— **This paper is an initial approach to create a controller for the game TORCS by learning how another controller or humans play the game. We used data obtained from two controllers and from one human player. The first controller is the winner of the WCCI 2008 Simulated Car Racing Competition, and the second one is a hand coded controller that performs a complete lap in all tracks. First, each kind of controller is imitated separately, then a mix of the data is used to create new controllers. The imitation is performed by means of training a feed forward neural network with the data, using the backpropagation algorithm for learning.**

I. Introduction

Video games are becoming more important in the present society, in a consumer product, as well as an opportunity for researching in AI (artificial intelligence). But, every video game player knows that the current AI in the games is very far from the human intelligence. When we are playing a game versus one or more NPC (non-player character) we realize very fast we are not playing versus another human and we find the way to beat the NPC, then the game becomes boring. Some ways to avoid this is by creating AI that cheats or playing in Internet against other human players, but this is also a problem because with a lot cheats or playing versus experienced human makes you lose in every game and the game becomes boring as well.

The first step for the AI should be to create opponents as intelligent as a human player, this will be a killer application [1]. But it is not the only step, we have to bear in mind that this AI must be able to adapt its behavior depending on the opponent, that is, it has to play in the same level of the human, neither better nor worse. In this way the AI will provide a better entertainment for the player.

This paper aims to be a starting point of our work to create competitive NPCs that imitates the human behavior. Learning a behavior by imitation does not only mean that we can create a NPC that plays like a human, but we can create a controller that can play as well as its opponent, in the same level. This would mean that the NPC player can adapt its behaviour to play like the human it is playing against and adapt its behaviour when the human improve his/her player skills to remain competitive.

For our goals we have selected a realistic enough game where a human plays versus one or more NPC and let us compare the results with other researchers. The game is TORCS (The Open Racing Car Simulator [1], see Figure 1), an open car racing simulator that is being used in several competitions of AI. This video game avoids the problem of complex behaviours like planning, we focus our research in behaviours that take place in a short period of time.

J. Muñoz, G. Gutierrez, A. Sanchis are with the Computer Science Department, University Carlos III of Madrid, Avda. de la Universidad 30, 28911 Leganés, Spain (emails: { jmfuente, ggutierr, masm }@inf.uc3m.es).

[1]The Open Racing Car Simulator - http://torcs.sourceforge.net

Fig. 1. Screenshot of TORCS

We have used three different controllers in our experiments, whose behavior has been learned. The first controller is a human player that drives the car through the middle of the road, the second one is a controller created by machine learning which won a TORCS competition in 2008, and the third one is a controller programmed by hand that performs a complete lap in all the tracks of TORCS. In all the experiments the controller created is a feed-forward ANN [2] (Artificial Neural Networks) that was trained with data generated by the controllers. The learning algorithm for the ANN was backpropagation.

In the next sections we will see first some related work of competitive AI in video games, specifically in racing games as TORCS, adaptive AI in video games, and behavior cloning. Next we will describe the domain we have selected to achieve our objectives, the competition of TORCS. Then, we will talk about the controllers used in the domain that tried to imitate. In the next section we will describe the process we have employed to create a controller that imitates other and the experiments done. Then, we will show the results of the experiments and finally we will conclude with some remarks and the future lines of research.

II. Related work

Creating NPC players could be a hard task. As the complexity and realism of the games grew, to program a NPC is harder, and it means more time and more money to develop the game. So the trend is to create video games where you can play versus other human players through Internet, or use AI techniques to create automatically NPC players.

978-1-4244-4815-9/09/$25.00 ©2009 IEEE

One wide area of researching is to create computational intelligence in games is related to the ANN or neuroevolution [3]. One example of this is NEAT (NeuroEvolution of Augmenting Topologies) [4], an effective method to create an ANN from scratch. NEAT starts with a small population of random ANN with a very simple topology, i.e. a fully connected network with only the input and the output layers, that evolves increasing its complexity and adaptation to the problem. Some games where NEAT has been applied successfully are NERO [5] and more recently TORCS [6]. Indeed, we have used one controller created with NEAT in our experiments that we will explain below.

One problem of NEAT is that the ANN created only take into account one objective, the fitness value, and in some cases this could be not enough. So some authors have added multi-objective to the neuroevolution [7] creating a method that can generate more than one NPC. All the NPCs created with this method have diverse behaviors that are adapted to the different objectives, this adds the possibility of showing multiple behaviors while playing. Even swap the behavior of the NPC when required.

Neuro evolution is not the only technique used to create the AI in video games. For example, reinforcement learning was used to create a multi-purpose bot for a FPS (first person shooter) video game [8]. A cognitive architecture, ICARUS, was also applied to create believable bots for FPS [9]. More related to racing games an ant colony optimization algorithm was used to improve a controller [10], genetic programming was used to evolve a controller [11] and genetic programming with multobjective evolution was used to create diverse opponents [7].

As we have said before, creating an intelligent NPC should not be the only goal of the AI. It is also needed to create opponents that suppose a challenge for the human player, this means that the AI of the games should be able to change its behavior to adapt it to the player. In [12] the authors use an approach called 'rapidly adaptive game AI' to adapt the AI of the game to the player, specifically the method applies continuously small adaptations to the AI based on the observations and evaluation of the user actions. Another example of adaptation is the Dynamic Scripting [13], this technique is based on a set of rules that are used for the game, whose weights to select one or another rule are modified through a machine learning algorithm. Dynamic Scripting was successfully applied to a commercial game called Neverwinter Nights.

Some researchers focus their research in creating interesting opponents instead of intelligent or adaptive ones. In [14] the author evolved neural-controller opponents that made the game more fun, that is, the evolutionary mechanism evolved the opponents while were playing against the player to make the game more fun instead of create optimal opponents that always win.

Another way to create NPC is by means of the imitation human behaviors [15]. The NPC player created in this way would show the same intelligence that the human it is imitating. A research field where imitation has been applied is the RoboCup, a simulated league of soccer. In [16] the authors clone the behavior of a RoboCup player using case base reasoning. Another example is [17] and [18] where the authors program robosoccer agents by modelling human behaviors with successful results. A sort of games where the imitation has also been applied are the FPS [19], in [20] the NPC learnt to imitate a human in the combat mode, that is the weapon that must be handling or how to shot in a realistic way. There is also some research in imitating more high-level human behavior from observation, FAMTILE [21] is an architecture used in a strategic game that tries to learn the context of the game and how it changes and then execute the appropriate actions for that context.

III. TORCS COMPETITION

The TORCS game has some features that make it perfect for researching. First, the game is a very realistic simulator that has a sophisticated physic engine that takes into account many aspects of the racing such as fuel consumption, collisions or traction. TORCS also provides a lot of tracks, cars with different features and several controllers for the cars. The last important feature is that TORCS is open software and that allows the researchers to make modifications to the game and adapt it to their requirements.

The advantage of using a competition as a benchmarking of our experiments is that we can compare our results with other researchers. So we have selected the Simulated Car Racing Competition [22] that takes place in some congress as IEEE Congress on Evolutionary computation (CEC), Computational Intelligence and Games Symposium (CIG) or IEEE World Congress on computation Intelligence (WCCI).

In the competition the TORCS game has been modified to allow external programs run as controllers. A client controller connects with a server bot in the TORCS game, the server sends to the client information about the status of the car and the client sends to the server the actions that must be executed. With this client-server architecture each programmer can program a controller in his/her preferred language and then connect it with the game. Another advantage of the architecture is that all the clients use the same information to create the controller because the sensors information sent to the clients is the same for all and no one can use information relative to the track or other internal information that TORCS manage.

The information provided by the server is related with the lap (current lap time, best lap time, distance raced, race position), the status of the car (damage, fuel, actual gear, speed, lateral speed and R.P.M.), the distance between the car and the track edges, the distance between the car and the opponents and the spin of the wheels. The actions that can be sent to the server to control the car are the acceleration level, brake level, the gear and the steering of the wheel. For more detailed information about the sensors and the effectors see [22].

IV. CONTROLLERS

In our experiments we use the data obtained from three different controllers: a human player, the winner of the WCCI 2008 competition and a hand coded controller. These controllers will be described below.

A. Human player

Due to the information that the human perceives from the game watching the monitor is much richer, much more than the other controllers manage, instead of the human was driving the car in a competitive way, he drove the car ignoring some information like how is the next curve if it is to the left or to the right, or how close is it. Thus, the human player drove the car trying to go through the middle of the road, with soft accelerations and brakes, braking much before the next curve started and performing the curves with a moderate speed without fast and sharp turns. The human tried to drive the car as a programmed controller would do, but with the mistakes that human makes.

B. NEAT controller

The second controller used is one created by Matt Simmerson by means of NEAT (a detailed description can be found in [22]). This controller was the winner of the WCCI 2008 Simulated Car Racing Competition.

As inputs of the ANN created by NEAT the author selected the current speed, the angle to track axis, the track position with respect to left and right edges, the current gear selection, the four wheels spin sensors, the current R.P.M and the 19 track sensors. All these inputs were scaled to the range [0,1]. The outputs of the ANN were the power (accelerate and brake), the gear change and the steering. The two first are in range [0,1] and the last one is in range [-1,1].

The fitness function used to evaluate the ANN in NEAT took into account the distance raced, the R.P.M, the maximum speed reached and a value to measure of how much the car stayed on the track.

Due to some restrictions of TORCS the controller only was trained in one track. The track selected contained different type of curves, to left and right, and also straights of varying lengths.

C. Hand coded controller

The idea of creating another controller was due to the human controller sometimes makes mistakes and the Simmerson's controller does not perform one complete lap in all the tracks and sometimes gets out from the track. Thus, a hand coded controller was created with these two requirements:

- to have the same outputs for the same inputs (does not make mistakes, tries to be deterministic)
- to perform a lap without getting out of the track, although the speed was not too high

To calculate the values for the acceleration and the brake level, this controller calculate the speed that the car should have, we called it the estimated speed. This is made by means of the information obtained from the three front sensors, which give the distance between the car and the edge of the track. If the distance is large then the speed should be high and if the distance is small the speed should be small too (See Equation (1), where $sum\_sensors$ is the sum of the three sensors and α and β are predefined parameters). With this value we calculated the difference (See Equation (2)) between the actual speed and the estimated speed. If the value is greater than 0 then the car should accelerate and if the value is lesser than 0 then the car must brake. The acceleration and brake values are proportional to the absolute value of the difference of the actual speed and the estimated speed (See Equations (3) and (4), where γ and δ are adjustment parameters).

$$\text{estimated speed} = \alpha \cdot sum\_sensors + \beta \tag{1}$$

$$\text{difference} = \text{estimated speed} - \text{actual speed} \tag{2}$$

$$\text{accelerate} = \frac{\text{difference}}{\text{estimated speed}} \cdot \gamma + 0.4 \tag{3}$$

$$\text{brake} = \frac{\text{difference}}{\text{estimated speed}} \cdot \delta \tag{4}$$

The steering value is slightly more complicated to calculate than the brake and accelerate values. First, we have to take into account whether the car is in a straight or in a curve. We suppose that the car is in a straight when any of the three front sensors has the maximum value, and in a curve otherwise. In this case, the steering is calculated as the difference of two of the three front sensors plus the angle with the axis (See Equation (5), where dif_sens is the difference between the two front sensors, $axis$ is the angle between the car and the track and ζ and η are adjustment parameters). If the car is in a curve then the steering is calculated in a similar way but more track sensors are taken into account (See Equation (6)), where sum_dif_sens is the summation of the difference of the track sensors, θ is an adjustment parameter, difference is the difference between the actual speed and the estimated speed and the other values are as in Equation (5))).

$$\text{steering} = \zeta \cdot dif\_sens + \eta \cdot axis \tag{5}$$

$$\text{steering} = \theta \cdot sum\_dif\_sens \cdot \text{difference} + \eta \cdot axis \tag{6}$$

Finally the gear is calculated rounding the result of the Equation (7), where λ is an adjustment parameter and $speed$ is the current speed of the car. For the gear the controller does not allow to change the gear twice in less than a second.

$$\text{gear} = \lambda \cdot speed \tag{7}$$

V. CONTROLLER LEARNING BY IMITATION

For our goal of learning the behavior of the controller we have used an ANN. The first step is to obtain the data from the controller we want to imitate, then the ANN is trained with the backpropagation algorithm and finally the new controller is tested in the tracks.

Instead of using all the data that the server of TORCS provides us as the inputs of the ANN, we have selected only some of them. So the inputs of the ANN are 28 and all are scaled into the range [0,1]:

- current speed of the car
- angle of the car with the axis
- the current gear
- the lateral speed of the car
- the R.P.M.
- the four spins of the wheels
- 19 sensor values with the distance between the car and the edges of the track

For the outputs, the data that must be sent to the server from the client to control the car, we have used three, all in the range [0,1]:

- the accelerate/brake value
- the gear
- the steering

The first output, the accelerate/brake value, is used for the acceleration and the braking. When this value is higher than 0.5 that means the car must accelerate, so the brake value is set to 0 and the accelerate/brake value is scaled from the range [0.5,1] to the range [0,1] to set the acceleration. When the value is lesser than 0.5 then the car must brake, so the acceleration value is set to 0 and the accelerate/brake value is scaled from the range [0,0.5] to the range [0,1] to set the brake.

For all the experiments the ANN has 3 hidden layers of 28 neurons each one, were trained during 1000 cycles and the learning rate was variable starting in 0.9 and finishing in 0.0001. We use the data of 17 road tracks (See Figure 2), but only if the controller complete almost 3 laps. More parameters were proved to train a controller for some tracks, we chose these experimental setup because we get good results with a good convergence in the error of the ANN.

Table I shows the number of patterns that have been obtained per each controller.

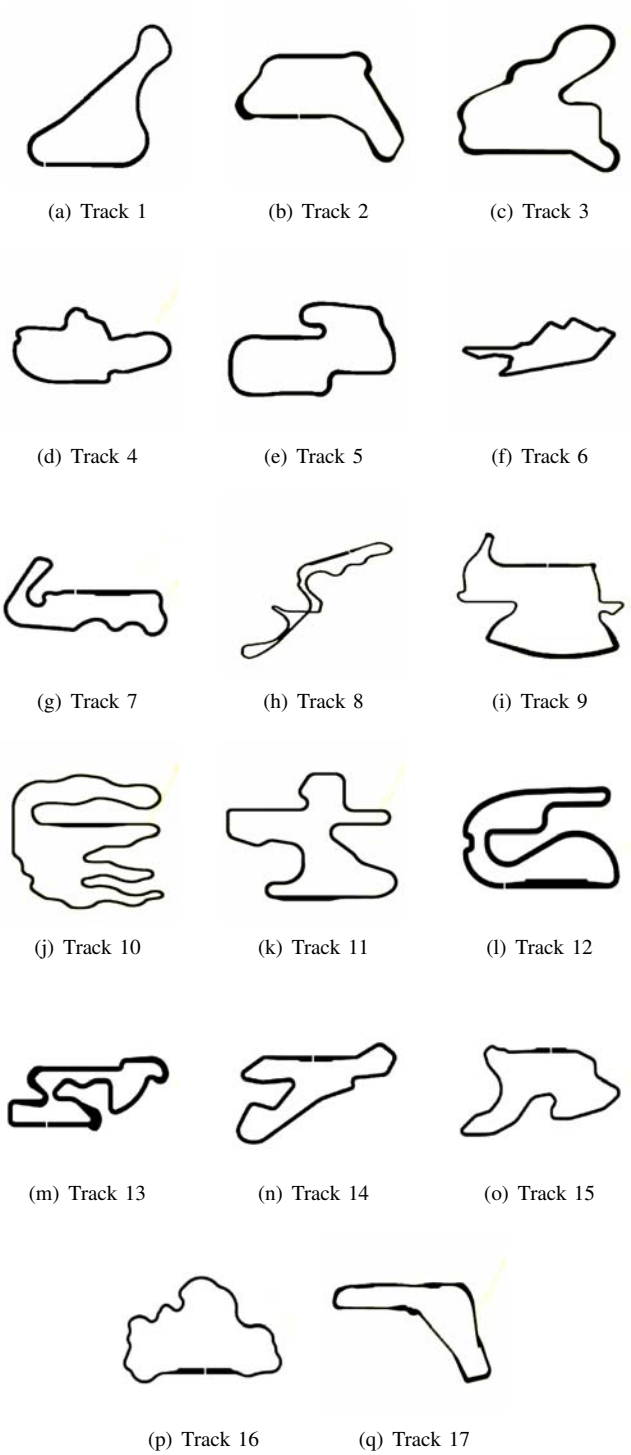
TABLE I

NUMBER OF PATTERNS

controller	number of patterns
human	257908
Simmerson	232251
hand coded	363853

VI. RESULTS

This section shows the results of all the controllers. For the controllers learning by imitation we used the data of all tracks to train the ANN of the controller and then test it in each track. Although we test the controllers in the same tracks where we get the data to train the ANN, we have to bear in mind that the neural network does not learn perfectly the patterns and with the same input the ANN gets a different output. This means that the simulation will be different, so



(a) Track 1 (b) Track 2 (c) Track 3

(d) Track 4 (e) Track 5 (f) Track 6

(g) Track 7 (h) Track 8 (i) Track 9

(j) Track 10 (k) Track 11 (l) Track 12

(m) Track 13 (n) Track 14 (o) Track 15

(p) Track 16 (q) Track 17

Fig. 2. Tracks used: 2(a) CG Speedway number 1, 2(b) CG Track 2, 2(c) CG Track 3, 2(d) Olethros, 2(e) Ruudskogen, 2(f) Street 1, 2(g) Wheel 1, 2(h) Wheel 2, 2(i) Aalborg, 2(j) Alpine 1, 2(k) Alpine 2, 2(l) E Track 1, 2(m) E Track 2, 2(n) E Track 3, 2(o) E Track 6, 2(p) E road, 2(q) Forza

the learned controller will not have the same inputs as the others controllers.

Table II, Table III and Table IV show the times obtanined by the controllers described before in each track: the human controller, the Simmeron's controller and the hand coded controller. Table V, Table VI and Table VII show the results of the learnt controllers for the human, Simmerson's and hand coded controllers. Table VIII shows the results of a controller created with the data of Simmeron's controller and human controller, Table IX shows the results of a controller created with the human and hand coded controllers and finally Table X shows the results of a controller created with the Simmeron's controller and the hand coded controller. First column of the tables (track) is the number of the track (See Figure 2). Second column (total time) is the total time needed to complete the number of laps (third column). The format of the time is $\langle minutes \rangle$: $\langle seconds \rangle . \langle tenths \rangle$. Fourth column (top speed) is the top speed reached by the controller in a lap. And the last column, damages, are the damages suffered by the car when it goes out of the track or changes the gear with high R.P.M. Some controllers do not complete a lap, in those cases the data of the column is shown in italic font, between brackets and the data shown is different: the total time column shows the time that the car stayed in the track, the lap column shows a decimal number with the proportion of the track that has been completed, the best lap column shows an estimation of the time the car would need to complete a lap and the last two columns show nothing.

For all the experiments where a controller has been tried to be imitated we realized that the gear value was not learned properly. Thus, the gear change was made by coded and the ANN output for the gear was not take into account. The Tables V, VI, VII, VIII, IX and X show the data with this modification in the controllers.

We can see that the Simmerson's controller is the only one of the three controllers that does not complete a lap in some tracks. The human and the hand coded control perform the 3 laps in all the tracks.

If we compare the human controller (Table II) and the imitated human controller (Table V) we can see that the human complete all tracks while the human imitated controller only performs more than a half of a lap in one track, but never a complete lap. The estimated best laps times for the imitated human controller are greater than the best lap times of the human except for the track which it is completed more than a half of a lap.

For the Simmeron's controller (Table III) and the imitated Simmerson's controller (Table VI) the results are more similar. The imitated controller here does not complete all the tracks that the Simmerson's controller does, but the best lap time is very similar in both controllers. We have to remark here that the top speed of the imitated controller is greater than the Simmerson's controller and that could be the reason that makes the cars go out of the track and not complete a lap.

TABLE II
HUMAN CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	02:15.34	3	00:42.69	234	0
2	03:08.56	3	00:59.58	263	0
3	03:50.98	3	01:15.07	223	239
4	06:55.40	3	02:12.05	291	2961
5	03:35.74	3	01:16.38	242	0
6	04:39.00	3	01:29.23	270	0
7	04:42.22	3	01:31.48	262	0
8	07:04.71	3	02:16.92	261	1
9	04:23.71	3	01:25.25	229	0
10	07:59.63	3	02:36.17	246	592
11	05:35.28	3	01:49.26	232	14
12	03:52.13	3	01:16.34	252	5
13	04:48.03	3	01:33.30	222	0
14	05:03.90	3	01:38.40	274	0
15	05:04.61	3	01:40.58	254	4
16	03:52.74	3	01:14.73	257	0
17	04:59.64	3	01:37.66	289	0

TABLE III
SIMMERSON CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	02:27.08	3	00:47.68	178	0
2	03:37.63	3	01:10.97	181	0
3	03:56.37	3	01:17.66	173	29
4	08:24.52	3	02:43.25	161	623
5	04:15.91	3	01:23.74	164	0
6	06:07.52	3	01:54.30	177	353
7	05:09.56	3	01:40.86	187	0
8	*[01:00.00]*	*[0.37]*	*[02:41.87]*		
9	04:59.90	3	01:36.21	160	6353
10	08:35.19	3	02:50.21	182	273
11	07:00.64	3	02:13.18	160	2362
12	*[00:32.00]*	*[0.37]*	*[01:26.48]*		
13	*[02:20.00]*	*[1.45]*	*[01:36.62]*		
14	05:37.83	3	01:50.94	197	0
15	*[01:12.00]*	*[0.63]*	*[01:53.91]*		
16	04:24.12	3	01:26.56	186	0
17	06:50.26	3	02:14.94	165	0

TABLE IV
HAND CODED CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	02:41.91	3	00:52.63	215	0
2	03:57.27	3	01:17.26	238	0
3	06:32.10	3	02:09.52	203	0
4	12:14.99	3	04:02.83	244	0
5	06:06.60	3	02:00.54	196	0
6	05:43.85	3	01:52.88	258	0
7	05:59.36	3	01:54.35	253	107
8	09:41.70	3	03:11.52	253	0
9	06:20.38	3	02:05.16	193	0
10	12:17.65	3	04:04.20	232	0
11	08:39.68	3	02:51.37	221	0
12	04:44.75	3	01:33.11	239	0
13	06:34.75	3	02:10.08	210	0
14	05:23.85	3	01:45.79	267	0
15	06:45.80	3	02:13.25	241	0
16	05:39.67	3	01:51.31	233	0
17	07:19.79	3	02:24.10	283	0

TABLE V

HUMAN IMITATED CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	*[00:23.00]*	*[0.35]*	*[01:06.35]*		
2	*[00:17.00]*	*[0.24]*	*[01:09.86]*		
3	*[00:10.50]*	*[0.11]*	*[01:39.51]*		
4	*[00:14.00]*	*[0.09]*	*[02:34.29]*		
5	*[00:10.40]*	*[0.11]*	*[01:37.28]*		
6	*[01:10.00]*	*[0.41]*	*[02:49.37]*		
7	*[00:17.00]*	*[0.18]*	*[01:32.78]*		
8	*[00:17.00]*	*[0.09]*	*[03:08.37]*		
9	*[00:09.00]*	*[0.08]*	*[01:48.29]*		
10	*[01:50.00]*	*[0.31]*	*[05:51.28]*		
11	*[01:39.00]*	*[0.94]*	*[01:45.52]*		
12	*[00:11.00]*	*[0.11]*	*[01:41.92]*		
13	*[00:09.50]*	*[0.1]*	*[01:39.66]*		
14	*[00:12.00]*	*[0.09]*	*[02:13.93]*		
15	*[00:13.50]*	*[0.11]*	*[01:57.56]*		
16	*[00:13.00]*	*[0.15]*	*[01:28.29]*		
17	*[00:22.00]*	*[0.21]*	*[01:46.93]*		

TABLE VI

SIMMERSON IMITATED CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	02:26.43	3	00:47.45	178	0
2	03:36.34	3	01:10.56	182	0
3	03:54.25	3	01:16.60	174	190
4	08:12.44	3	02:39.97	161	722
5	*[00:54.00]*	*[0.59]*	*[01:31.13]*		
6	*[01:17.00]*	*[0.75]*	*[01:43.29]*		
7	05:03.98	3	01:39.58	192	0
8	*[00:58.00]*	*[0.37]*	*[02:37.85]*		
9	*[02:06.00]*	*[0.86]*	*[02:25.97]*		
10	08:46.89	3	02:48.47	186	460
11	06:47.36	3	02:07.30	164	2738
12	*[00:33.00]*	*[0.39]*	*[01:24.94]*		
13	*[00:45.00]*	*[0.47]*	*[01:36.34]*		
14	05:32.12	3	01:48.83	208	23
15	*[01:12.00]*	*[0.63]*	*[01:55.02]*		
16	04:23.94	2	01:26.55	186	0
17	*[01:07.00]*	*[0.48]*	*[02:19.90]*		

TABLE VII

HAND CODED IMITATED CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	02:42.26	3	00:52.16	219	0
2	03:59.88	3	01:18.01	224	0
3	06:31.10	3	02:08.67	199	0
4	*[00:56.00]*	*[0.22]*	*[04:14.55]*		
5	06:03.19	3	01:59.18	201	0
6	05:46.23	3	01:53.50	227	0
7	07:53.25	3	02:07.88	227	620
8	09:44.15	3	03:14.32	227	0
9	06:14.62	3	02:02.88	201	0
10	12:17.18	3	04:04.05	223	0
11	08:35.30	3	02:50.16	213	0
12	05:01.56	3	01:38.29	223	668
13	06:21.47	3	02:05.52	221	0
14	05:45.10	3	01:49.87	256	240
15	06:48.46	3	02:14.27	217	0
16	05:43.09	3	01:52.96	231	0
17	08:05.94	3	02:38.60	215	0

TABLE VIII

SIMMERSON-HUMAN IMITATED CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	*[00:23.50]*	*[0.47]*	*[00:49.58]*		
2	*[01:21.50]*	*[1.23]*	*[01:06.39]*		
3	*[42:00.00]*	*[0.46]*	*[30:27.55]*		
4	*[00:17.50]*	*[0.11]*	*[02:41.67]*		
5	*[00:11.50]*	*[0.09]*	*[02:01.45]*		
6	*[01:10.50]*	*[0.75]*	*[01:34.24]*		
7	*[00:44.00]*	*[0.41]*	*[01:48.27]*		
8	*[00:29.00]*	*[0.15]*	*[03:07.44]*		
9	*[00:11.00]*	*[0.09]*	*[02:04.81]*		
10	*[01:35.00]*	*[0.45]*	*[03:29.26]*		
11	*[00:24.00]*	*[0.16]*	*[02:33.48]*		
12	*[00:50.00]*	*[0.37]*	*[02:14.01]*		
13	*[00:46.00]*	*[0.45]*	*[01:41.52]*		
14	*[01:06.00]*	*[0.6]*	*[01:49.98]*		
15	*[00:40.00]*	*[0.37]*	*[01:48.65]*		
16	*[00:54.00]*	*[0.53]*	*[01:41.17]*		
17	*[00:50.00]*	*[0.36]*	*[02:19.04]*		

TABLE IX

HAND CODED-HUMAN IMITATED CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	*[00:14.00]*	*[0.23]*	*[01:00.00]*		
2	*[00:23.00]*	*[0.28]*	*[01:22.68]*		
3	*[00:12.00]*	*[0.11]*	*[01:53.72]*		
4	*[01:20.00]*	*[0.44]*	*[03:02.09]*		
5	*[00:16.00]*	*[0.11]*	*[02:29.67]*		
6	*[00:30.00]*	*[0.25]*	*[01:59.47]*		
7	*[00:19.00]*	*[0.19]*	*[01:39.49]*		
8	*[00:12.00]*	*[0.06]*	*[03:26.83]*		
9	*[00:14.00]*	*[0.09]*	*[02:40.97]*		
10	*[00:20.00]*	*[0.07]*	*[04:30.43]*		
11	*[00:50.00]*	*[0.45]*	*[01:51.63]*		
12	*[00:30.00]*	*[0.28]*	*[01:48.10]*		
13	*[00:10.50]*	*[0.08]*	*[02:07.09]*		
14	*[00:50.00]*	*[0.31]*	*[02:42.68]*		
15	*[00:19.50]*	*[0.12]*	*[02:37.45]*		
16	*[00:27.50]*	*[0.2]*	*[02:15.83]*		
17	*[00:41.00]*	*[0.23]*	*[02:58.30]*		

TABLE X

SIMMERSON-HAND CODED IMITATED CONTROLLER RESULTS

Track	Total time	Laps	Best lap	Top speed	Damages
1	02:27.62	3	00:47.45	178	0
2	03:37.59	3	01:10.70	182	0
3	*[03:24.00]*	*[1.72]*	*[01:58.80]*		
4	*[04:30.00]*	*[1.59]*	*[02:50.02]*		
5	*[01:00.00]*	*[0.61]*	*[01:38.71]*		
6	*[01:29.00]*	*[0.75]*	*[01:58.14]*		
7	05:08.45	3	01:40.89	192	0
8	*[01:04.00]*	*[0.38]*	*[02:50.44]*		
9	05:47.86	3	01:45.93	161	7585
10	*[01:40.00]*	*[0.51]*	*[03:17.36]*		
11	06:05.49	3	01:59.56	163	3136
12	*[00:34.00]*	*[0.38]*	*[01:28.56]*		
13	05:33.20	3	01:35.55	173	4276
14	*[00:39.00]*	*[0.29]*	*[02:14.45]*		
15	*[03:28.00]*	*[1.65]*	*[02:06.19]*		
16	04:27.26	3	01:27.25	186	0
17	*[01:20.00]*	*[0.49]*	*[02:43.74]*		

The hand coded controller (Table IV) and the imitated hand coded controller (Table VII) are the controllers that perform more complete laps in all tracks, but they are also the slowest controllers. The only track that is not completed by the imitated hand coded controller is because the car jumps with a bump and losses the control crashing against the wall without recovering itself. In this case, the imitated controller has the same best time lap than the hand coded controller, but with more damage in some tracks. The speed in these two controllers is different too, the imitated controller has more top speed in some tracks and less in others.

The controller created with data from the human and the Simmeron's controller (Table VIII) does not complete a lap like the imitated human controller. But the percentage of completed laps in this controller is higher than in the others. Something similar happens with the controller created from the human and Simmeron's controller data, the car does not complete a lap and the percentage of completed laps is higher than the imitated human controller. The last of the three mixed controllers, the car that has been created with the hand coded and Simmerson's data (Table X), complete a lap in more than the half of the tracks. The results of this last controller are a mix between the hand coded controller and the Simmeron's controller, although more similar to the imitated Simmeron's controller. It is also remarkable that there are some tracks that the Simmeron's controller does not perform and this last controller does.

A controller created with the data of the human, Simmersons and handcoded controllers was not created due to we did not get good results with mixed configurations, as we show in Table VIII, IX and X.

VII. Conclusions

The first conclusion that we can remark from the results we got is that it is very complicated to learn the human behavior in a video game. This is due to several causes: neither the human makes the same action in the same circumstances, nor performs all actions in the proper way. The human makes a lot of mistakes that have to resolve. This sort of data makes completely impossible that an ANN could learn something useful. Another problem is that humans use more information than we provided to the ANN network.

We have probed that it is possible to learn a behavior from non-human controllers. The data obtained from this sort of controllers do not have the same problems as the human data, so an ANN can learn its behaviour. But there is also one problem related to this controller and it is that the gear change has not been learned, despite of being probably the easiest output to learn. Maybe it is due to the high amount of data and due to the gear is also an input of the ANN.

In the experiments where we have used mixed data from two types of controllers the results show that those controllers do not work. The controller learned has mixed features of both the controllers that have been used, but no one of these features is learned properly, so the car goes out of the track easily and the simulation ends.

VIII. Future works

There are some research lines that must be done in the future to improve the results of the controllers. First step should be performed is a pre-process of the data before use it to train the neural network. Two ideas must be borne in mind:

- the amount of data must be decreased removing duplicated or very similar data
- the data patterns with the same input but different output must be removed to avoid a wrong learning process

Another idea is to train the controllers with some data of one controller. For example, the data of the straights of one controller that performs the straights good and the data of the curves of another controller that make well the turns.

One step to perform if we want to be able to imitate the human behavior is increase the information used to train the ANN. That is, there is a lack of information in the data to create a controller that imitates a human because the human player senses more information from the domain than the other controllers and the human also can remember and improve his/her behavior in each lap. Another problem with the data obtained from the human player is that he is not perfect and does not perform the same action under same circumstances.

The ANN is another point that must be improved. With the current ANN there is not context information, the controller does not remember its past actions and can not take a decission with that information. Thus, it could be a good idea to use recurrent neural networks and test them with different learning algorithms like backpropagation through time [23] or RTRL (Real Time Recurrent Learning) [24].

Acknowledgment

This work was supported in part by the University Carlos III of Madrid under grant PIF UC3M01-0809 and by the Ministry of Science and Innovation under project TRA2007-67374-C02-02.

References

[1] J. Laird and M. Van Lent, "Human-level AI's killer application," *AI magazine*, vol. 22, no. 2, 2001.

[2] X. Yang and J. Zheng, "Artificial neural networks," *Handbook of Research on Geoinformatics*, p. 122, 2009.

[3] R. Miikkulainen, B. Bryant, R. Cornelius, I. Karpov, K. Stanley, and C. Yong, "Computational intelligence in games," *Computational Intelligence: Principles and Practice. Piscataway, NJ: IEEE Computational Intelligence Society*, pp. 155–191, 2006.

[4] K. Stanley, *Efficient evolution of neural networks through complexification*. The University of Texas at Austin, 2004.

[5] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the NERO video game," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 653–668, 2005.

[6] 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.

[7] A. Agapitos, J. Togelius, S. Lucas, J. Schmidhuber, and A. Konstantinidis, "Generating diverse opponents with multiobjective evolution," in *Computational Intelligence and Games, 2008. CIG '09. IEEE Symposium On*, Dec. 2008, pp. 135–142.

[8] M. McPartland and M. Gallagher, "Creating a multi-purpose first person shooter bot with reinforcement learning," in *Computational Intelligence and Games, 2008. CIG '09. IEEE Symposium On*, Dec. 2008, pp. 143–150.

[9] D. Choi, T. Konik, N. Nejati, C. Park, and P. Langley, "A believable agent for first-person shooter games," in *Proceedings of the Third Annual Artificial Intelligence and Interactive Digital Entertainment Conference*, 2007, pp. 71–73.

[10] L. delaOssa, J. Gamez, and V. Lopez, "Improvement of a car racing controller by means of ant colony optimization algorithms," in *Computational Intelligence and Games, 2008. CIG '09. IEEE Symposium On*, Dec. 2008, pp. 365–371.

[11] A. Agapitos, J. Togelius, and S. Lucas, "Evolving controllers for simulated car racing using object oriented genetic programming," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM New York, NY, USA, 2007, pp. 1543–1550.

[12] S. Bakkes, P. Spronck, and J. van den Herik, "Rapid adaptation of video game ai," in *Computational Intelligence and Games, 2008. CIG '09. IEEE Symposium On*, Dec. 2008, pp. 79–86.

[13] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.

[14] G. Yannakakis and J. Hallam, "Evolving opponents for interesting interactive computer games," *From Animals to Animats*, vol. 8, pp. 499–508, 2004.

[15] C. Bauckhage, C. Thurau, and G. Sagerer, "Learning human-like opponent behavior for interactive computer games," *Lecture notes in computer science*, pp. 148–155, 2003.

[16] M. Floyd, B. Esfandiari, and K. Lam, "A Case-based Reasoning Approach to Imitating RoboCup Players," in *Proceedings of FLAIRS-2008, Florida AI Research Symposium*, 2008.

[17] R. Aler, O. Garcia, and J. Valls, "Correcting and improving imitation models of humans for robosoccer agents," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, Sept. 2005, pp. 2402–2409 Vol. 3.

[18] R. Aler, J. Valls, D. Camacho, and A. Lopez, "Programming Robosoccer agents by modeling human behavior," *Expert Systems with Applications*, vol. 36, no. 2P1, pp. 1850–1859, 2009.

[19] C. Thurau, C. Bauckhage, and G. Sagerer, "Imitation learning at all levels of game-ai," in *Proceedings of the international conference on computer games, artificial intelligence, design and education*, 2004, pp. 402–408.

[20] B. Gorman and M. Humphrys, "Imitative Learning of Combat Behaviours in First-Person Computer Games," in *Proceedings of the Tenth International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games (CGAMES'07)*, 2007.

[21] B. Stensrud and A. Gonzalez, "Discovery of High-Level Behavior From Observation of Human Performance in a Strategic Game," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 3, pp. 855–874, 2008.

[22] 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 '09. IEEE Symposium On*, Dec. 2008, pp. 119–126.

[23] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[24] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.