# Comparing the Performance of Finite-State Machines with Different Numbers of States on TORCS

Bruno H. F. Macedo
Gabriel F. P. Araujo
Gabriel S. Silva
Matheus C. Crestani
Yuri B. Galli
Guilherme N. Ramos
*University of Brasília*

## Abstract

This work presents two different approaches developed to control a self-driving car in the racing environment simulated by TORCS. The problematic is the understanding of the behavior that a pilot assumes during a race, and the proposals for dealing with it are two finite-state machines that reproduce this behavior. The evaluation of the methods proposed provided a comparison between finite state machines in which the one with fewer states led to improved performance in simulated races from TORCS.

**Keywords:** finite-state machine, computer games, TORCS, SCRC, artificial intelligence, genetic algorithm, self-driving car

**Author's Contact:**

{bhfmacedo}@gmail.com
gnramos@cic.unb.br

## 1 Introduction

Automation of day to day tasks is an endeavour that has moved a large amount of scientific resources in the recent history [Terzic et al. 2008] [Bejczy 2012]. One specific example is target of research around the globe by a lot of universities, companies and industries, which is the automation of vehicles, more specifically, automobiles. The objective of such attempts is the development of artificial intelligences capable of driving a car safely [Carbaugh et al. 1997], with traffic law enforcement, real-time decision making, efficiency and, in addition, resource economy - as with gas, pollution emission [Barth 2013] or even time. The practical applications of such controllers in autonomous vehicles are numerous, for example, the researches pursued by DARPA [1].

Practical systems are expensive to build effectively, which is the reason why simulations are used for solving this problem. Games present a well defined environment wich may simulate complex situations. More over they provide a large set of possibilities for comparing AI solutions for specific problems such as path planning [de Freitas et al. 2012], humam pose recognition [Shotton et al. 2011], and others

The Simulated Car Racing Championship (SCRC), using the platform TORCS (The Open Racing Car Simulator), has brought an excellent environment for benchmarking AI approaches for the problem of autonomous car controllers [Loiacono et al. 2010]. Even with the advent of computer simulations, finding the optimum behaviour of a car controller is a complex matter, so, many approaches have been suggested, such as heuristic algorithms [Quadflieg et al. 2014] and modular and fuzzy architectures [Onieva et al. 2012]. The strategy adopted in this work was to divide the problem into smaller portions, i.e., less complicated subproblems, in order to implement a finite-state machine that admittedly covers all necessary behaviours.

A finite state machine is a well-known algorithm used as solution in the literature in the attempt to solve several problems. For instance, a state machine was developed to control air volume in air conditioning systems [Bernaden 1999]. Another example is the surveillance systems in highways for detecting vehicles using a finite state machine [HyunKim et al. 2013]. To contextualize of this paper, previous controllers were designed using finite state machines, one of them was designed by Diego Perez, a controller which uses a state machine with fuzzy logic to solve the simulated car racing problem [Perez et al. 2009].

One way to enhance the performance of the controllers developed is by evaluating each and every possible set of parameters or configurations it could assume, but that choice is not always affordable due to large search spaces. Considering this, after an initial structure of the controller was designed, a method of computer-aided fine tuning was assimilated to it, which was a genetic algorithm.

The rest of this paper is structured as follows: Section 2 introduces TORCS, the working environment used in the problematic presented, along with the competition, SCR Championship, that currently represents the metric to evaluate the performance of controllers proposed for this environment; the section also presents what is already being done at this context in related works. Section 4 then explains the proposal of the two developed controllers, clarifying their behaviour and structure, and briefly describing how they were augmented by the computer-aided method of a genetic algorithm. Section 5 describes how the validation process occurred through the methodology, the experiments and the results achieved, including their correspondent analysis. Section 6 provides conclusions about the results acquired, which establish the comparison between the finite state machines with few and with moderate number of states, pointing out prospects about what might be done in future works to improve those results.

## 2 The Simulation Environment

Multiple applications that require verification and evaluation have become dependent of expensive resources, which makes their development a delicate matter. Examples of systems whose tests involve costly supplies are medical education [Zhang et al. 2007], the aviation industry [Duncan and Feterle 2000] and automotive research and development [Xue et al. 2011]; because naturally no company or enterprise has expendable numbers of bodies for tests, or airplanes and cars to crash in order to safe-proof and improve. In these conditions computer simulations arise, virtually reproducing the environments of the said and of other important applications so that their testbeds reduce the need of the scarce resources related to them.

Many other reasons justify the use of virtual simulations, varying from one application to another, and, because of this, countless companies are incorporating them in their educational and training process [Sim 2015] [Ope 2014]. A few number of reasons why a simulation was used in this work:

- managing control operations;
- planning actions;
- understanding a problem and finding out how to react to it;
- training and learning through experimentation.

Simulations represent just an approximated model of the systems they try to imitate, but in most cases the effects of the estimates adopted in the attempt to do this do not come to the point that the model becomes invalid. Similarly, car racing simulations also incorporate flawed modeling - regarding friction, gravity, trajectory lines, and so on - but resorting to them may save some spare parts that would most likely be jeopardized in real-life tests. So, new studies, new researches and new automation methods might be evaluated through simulations and avoid costs in the development process.

[1] http://www.darpa.mil/our-research

## 2.1 TORCS

The Open Racing Car Simulator (TORCS) is a platform that is renowned for its highly credible physics modeling engine and yet user-friendly interface for car racing simulations [Wymann et al. 2014]. It is a modern, multi-player and multi-agent car simulator; it also is an interface that is widely used for benchmarking AI [Loiacono et al. 2010] due its high degree of modularity and portability, concerning multi-platform environments and support to the programming languages C and C++. Artificially intelligent agents can be developed as modules inside TORCS where there are many possible levels of abstraction. For example, at the car level, there is an intelligent control system for each car component; at the driver level, mid-level control systems for complex driving agents could be implemented using the partial simulation information given by a low-level API provided by TORCS.

The TORCS engine [Wymann et al. 2015] uses a discrete-time simulation, whose discretization is set to approximately 0.002s of simulation time, and engine solves differential equations using Ëuler method, and all basic elements of the vehicular dynamics are handled, which are:

- basic properties of the vehicular system;
- mechanical details;
- dynamic and static friction;
- aerodynamic model.

Mass, rotational inertia of the car, engine, wheels, and other components, are included in the model of the vehicular system; while the types of different suspension, links, and differentials are done so in the mechanical model. The profiles for different ground types with both dynamic and static friction are also included; this way, the aerodynamics modeling includes slipstreaming and ground effects, that vary from one profile to another. Nevertheless, the simulation engine can be replaced or easily modified as a result of the modularity supplied by TORCS. The interface with this platform occurs by means of a sensor-based interaction system in which the developer is able to interpret received parameters of the car - such as speed in X, Y and even Z axes - and control the car through programming its actuators, some of which are acceleration and steering.

Another credibility factor for this platform is its non-punctual cars that interact with each other in the races by a life-like collision system. Still, TORCS is still a simulator, and its limitations, along with the defined racing environment and the modeled car, are more than likely to affect any results obtained. This is an inherited characteristic of any real-life problem simulation, what in academia is denominated *reality gap* [Whitton and Moseley 2012, ch. 8], and it stems from the simplifications made concerning the car models, the technical features of the tracks, and so forth.

The SCR interface occurs in a way that TORCS allows the controller to have a full view of the environment including its exact location inside the track, the geometry and friction and also the exact location of all the other cars. In order to separate the controller logic from the simulator, the SCR interface was designed [Loiacono et al. 2010]. This abstraction limits the information received by the controller, and the communication between it and TORCS is made through a client-server interface, with each player receiving information from the server regarding the sensors of the car, and in return providing actuator values that determine how the controller is supposed to drive the car.

## 2.2 The SCR Championship

The Simulated Car Racing Championship (SCRC) is an example of a well-known competition which utilizes TORCS as simulator [Cardamone et al. 2014]. Being an event joining three competitions held at major scientific conferences, such as *IEEE Congress on Evolutionary Computation* [2], *Genetic and Evolutionary Computation Conference* [3] and *IEEE Conference on Computational Intelligence and Games* [4], it is an accepted metric of evaluation in the fields of Evolutionary Computation and Computational Intelligence regarding games in general.

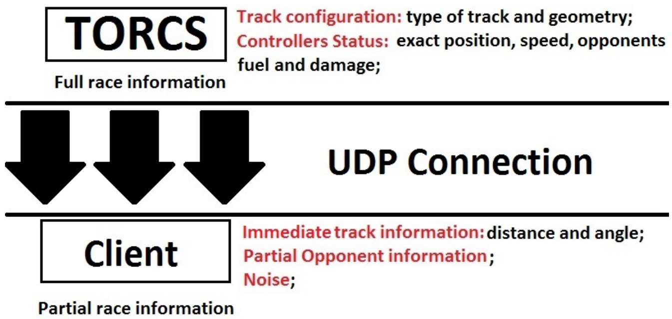


**Figure 1:** *Available data inside TORCS becoming data accessible to the client*

Race tracks are categorized into *Road*, *Dirt* and *Oval* inside TORCS. The races from the SCRC take place in track types decided by the organization of the championship, information which is not provided to the participants and that may incorporate maps that are unknown to them. The competition adopts a structure that gathers a *Warm-up* stage, a *Qualifier* stage and a *Final* race. Noise can be introduced in the sensors, option that is present during the actual competition. The complete sensorial input information and all the details concerning the race stages and types are presented at the Simulated Car Racing Championship Competition Software Manual [SCR 2013].

The reason why TORCS presents itself as a satisfactory AI benchmark, in combination with SCRC, is because even though there are multiple possibilities on how the sensorial input received from the server can be translated into the behavior of the actuators, they can all be compared in a race, which has a robust and steady scoring and evaluational system. In other words, there are many different approaches concerning how to teach the racer encoded by the developers to drive in a racing competition only with the information given by the sensors, and the metric to that issue is the performance on the race itself.

# 3 Related Works and State of the Art

It is very common among some of the SCRC awarded controllers the incorporation of machine learning in their driving methods, along with other evolving techniques using artificial intelligence [Loiacono et al. 2010]. Due to the uncertainty of the environment, track is initialy unknown to the controller, learning procedures may grant some advantage, enhance performance and competitiveness. Essentially, there are two ways of evolving learning solutions: Online Learning and Offline Learning.

According to Tom Mitchel in *Machine Learning* [Mitchel 1997]:

> *"Systems that learn by moving about the real environment and observing the results are typically called online systems, whereas those that earn solely by simulating actions within an internal model are called ofline systems."*

The current champion of the SCR Championship is the controller *Mr. Racer* [Quadflieg et al. 2014], and it has proven to be the State of the Art by winning the last three competitions that happened from 2011 to 2013. The authors of this implementation employ several heuristics and black-box optimization methods in order to reproduce the mechanisms to which human racing drivers resort, doing so by means of a modular structure. *Mr. Racer* uses a Covariance Matrix Adaptation Evolution Strategy (CMA-ES), to evolve parameters offline.

According to the founders of the competition [SCR 2013] and the authors of *Mr. Racer* themselves, *AUTOPIA* [Onieva et al. 2012] is

[2] http://www.cec2015.org/
[3] http://www.sigevo.org/gecco-2015/
[4] http://www.ieee-cig.org/

another competitive controller, with the potential to even be the best one available. *AUTOPIA* implements a modular Fuzzy Architecture, whose division contains gear, steering and speed control; and it is optimized by means of a genetic algorithm for Offline Learning, and by means of landmarking the lane exit points for further speed reduction for Online Learning.

These and other controller exemplifications [SCR 2013] served as criteria for the analysis and development of the approach presented in this paper. Elements incorporated and adapted from them feature modularity, Offline Learning through genetic algorithms, Online Learning through landmarking and choosing sets of parameters for different categories of tracks, etc.

# 4 The FSMDriver

According to Mat Buckland in *Programming Game AI By Example* [Buckland 2005]:

> *"A finite state machine is a device, or a model of a device, which has a finite number of states it can be in at any given time and can operate on input to either make transitions from one state to another or to cause an output or action to take place. A finite state machine can only be in one state at any moment in time."*

An architecture following this guideline was chosen in order to transform the problem of complex driving into smaller problems that describe the situations found within the racing environment.

## 4.1 The Design of the Behavior States

Initially, the design of the finite state machine proposed comprised the following states:

- *Straight Line*;
- *Approaching Curve*;
- *Curve*;
- *Out of Track*;
- *Stuck*.

Essentially, for this first method, normal behavior covered *Straight Line*, *Approaching Curve* and *Curve*, as the controller was located inside the track boundaries, whereas exception behavior consisted of *Out of Track* and *Stuck*, situations in which recovery actions are expected.

The main difference between the method described and the second one is the way they deal with normal behavior, while one separates it into *Straight Line*, *Approaching Curve* and *Curve*, the other treats it as a unique conduct in the form of the *Inside Track* state. Therefore, the modified finite-state machine was composed by only three states, which were:

- *Inside Track*;
- *Out of Track*;
- *Stuck*.

### 4.1.1 Normal Behavior

If a controller using the first method was currently in *Straight Line*, it would be expected of him to simply go as fast as he could, maitaining itself parallel to the track axis. When in *Approaching Curve* state, he would reposition himself and achieve a certain calculated target speed proportionally to the curvature of the approaching curve in order to achieve higher speeds once inside of it. For example, if there is a sharp left turn nearby, the controller sets a target steer that the car needs to obtain before entering it, while bringing the car further to the right of the lane; that way, when the left turn comes, the car can proceed with a less abrupt change in steering, which as a consequence results in a higher speed. Otherwise, when in situations of *Curve*, the pilot would stop braking while maintaining the steering direction towards the sensor pointing the biggest distance value read - which represents the direction of the curve, prepared by the approaching curve state. Figure 2 exhibits a car receiving information from the sensors, with the biggest vector being the direction of the curve to be entered.

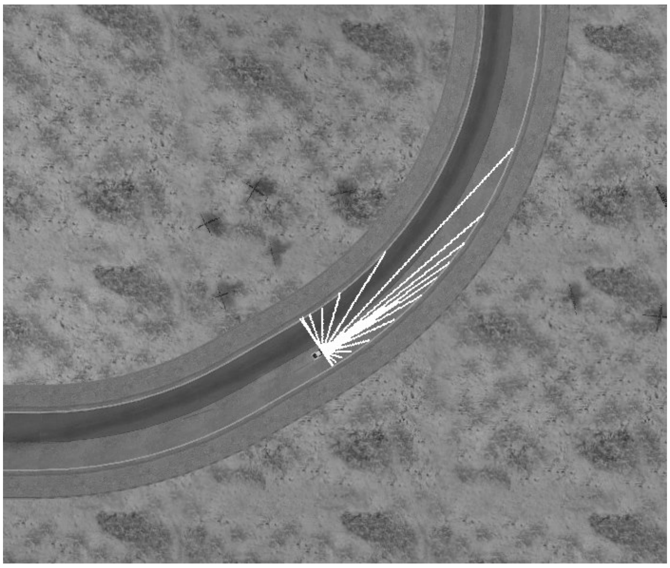

**Figure 2:** *Sensorial input indicating a curve through the biggest value received*

*Inside Track*, therefore, is how the car, desirably, will spend most part of the time, if the second method is being employed. The controller calculates a *target speed* based on how far the car is from the farthest edge of the track, then, it assumes a position of adjusting the speed until it reaches this velocity while driving towards the sensor with the biggest read value. The distance of this method conveys the greater length the car may advance with little or sometimes without significant steering changes. This state also brakes, if necessary, in situations that the car assumes a speed higher than the one calculated to be the target.

### 4.1.2 Exception Behavior

*Out of Track* is when, for any reason, the car is found outside of the track limits. In this case, the proper behavior is to try to return to the lane. Since there are many ways that a car may be out of the track, reentering it in an efficient way might require possible different angles as well, so, every time the car exceeds the inside boundaries of the lane, a proper returning angle is calculated. In road tracks, the outside track normally has a different terrain, sometimes dirt-based, meaning that skidding frequently occurs, and in an effort to avoid this, a control system to brake when the car begins skidding above a threshold was implemented. Figure 3 demonstrates a car returning from outside of the track with a certain angle.

Furthermore, the *Stuck* state represents any given situation that the car is unable to progress in the race. This is a delicate state, because it presents itself as difficult to identify and also due to its impact to the performance of the controller. In order to detect *Stuck* circumstances, the speed of the car is monitored throughout the race, during every game tick, if it lingers with a low speed for a determined period, then it is considered stranded, or stuck. This state activates the reverse gear of the car and turns it until its front is directed towards the correct axis of the track. The reason why *Stuck* is a sensitive state is because, when detected early, might indicate false positive, and, when detected late, could lessen the efficiency of the controller. Thusly, detecting *Stuck* situations is crucial, and so is handling the car out of them.

## 4.2 Five-State FSM

The first conceived method was named "Five-State FSM", naturally because it is a finite state machine with five states. The real first model of the finite-state machine did not have an *Approaching*

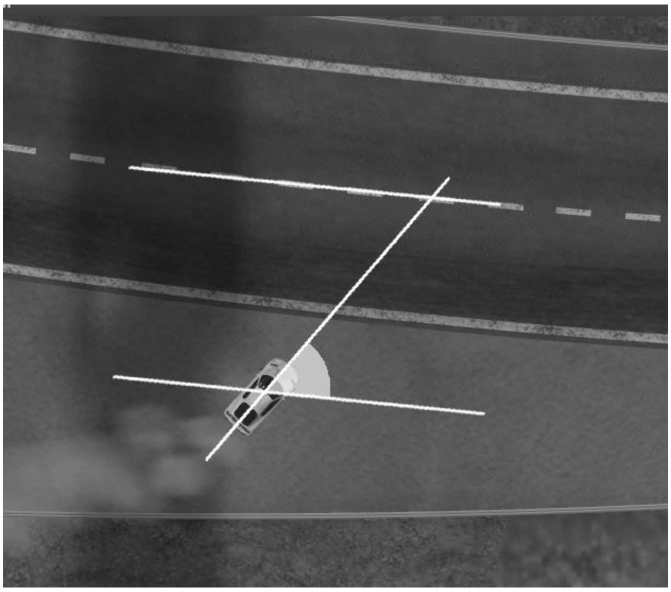

**Figure 3:** *Angle between car and track axis.*

*Curve* state, but, as the implementation of *Straight Line* and *Curve* were so different from one another, a preparation had to be established so as to smoothen the transitions between them. Figure 4 represents the states diagram of the Five-State FSM.

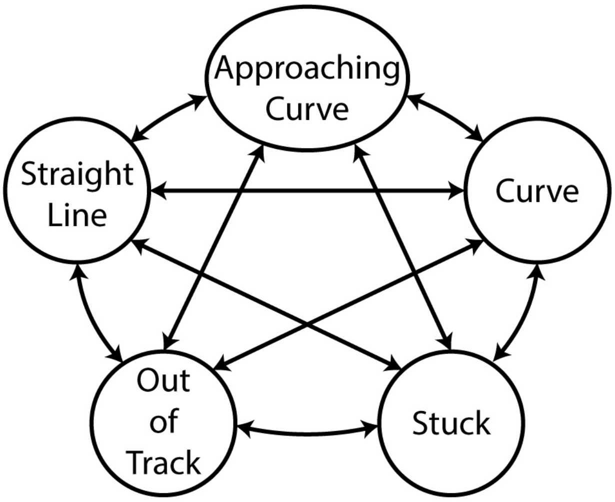


**Figure 4:** *States diagram of the Five-State FSM*

Figure 4 instantiates the relations between the states through two-headed arrows connecting a state to every other one, which is done because, from a particular state, the controller can assume any other in a following instant.

In order to decide wich state should be active at each moment a transition function was defined. It would take into account the covariance among the range finders. See Figure 2

One big problem about this approach is that the function responsible for choosing which state is more appropriate for each situation would more than often be overcharged, and, in some cases, rather different sets of parameters received by it would result in the same classification among the states. Thus, in order to minimize the dependency of the driving performance in relation to the function in charge of the transition between states, a project decision was made to reduce the number of states.

In addition, the angles - with relation to the car axis - of all the 19 range finder sensors were chosen in a manner that included as much information of the track as possible. For instance, if all the angles were to be initialized pointing only to one side of car, the data concerning the other side would be neglected. For the Three-State FSM, the angles were instantiated using steps of 10 degrees, which resulted in angles ranging from -90 to +90 degrees, 0 meaning the direction that indicates the front of the car.

### 4.3 Three-State FSM

The finite state machine with less states was designed from a derivation of the previous one, by adapting the normal behavior and simplifying it to form the denominated Three-State FSM. The exception behavior remained the same, since it should not affect the car as much if the adaptation was well performed. Figure 5 shows the modified states diagram of the Three-State FSM.

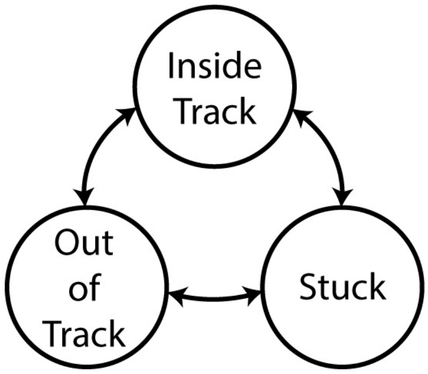


**Figure 5:** *States diagram of the Three-State FSM*

Just as the diagram of the Five-State FSM portrayed, there is no restriction of order concerning the states for the Three-State FSM. Figure 5 then displays this relation, which is the possibility of assuming an arbitrary state after any other.

The transition function in this architecture is simplier than the previous one. It only checks one range finder since it is very easy to say whether the car is inside or outside the track.

Besides the states, there is also a learning module that is called whenever the *Out of Track* state is requested. This module records both speed and position from the state of the car in the proximity of the departure from the track, and retaining these pieces of information allows the controller to slow down in subsequent laps when approaching the segments highlighted by the learning module. The implementation of this procedure consists on replacing the speed recorded from the landmarked position by a slower one on future occurrences.

In order to maximize the efficiency of the information received from the track, the vector of sensors in the Three-State FSM was initialized according to a normal distribution, i.e., the sensors are more densely distributed in front of car and less on the sides.

Each state in the two ways described to deal with the process becomes, ideally, an independent problem, whose solution can be attacked separately. This way, they can all have individual sets of parameters susceptible to improvement.

### 4.4 Search for Parameter Values - Genetic Algorithm

Due to the quantity of parameters to be tuned and the defined granularity, the search space becomes enormous and renders fundamental the use of a search algorithm. This technique optimizes the process of finding better configurations by being more incisive and saving resources such as computational time and space. In the present study, an evolutionary algorithm was chosen for this task.

Genetic Algorithms [Holland 1984] are evolutionary algorithms inspired by nature, in special by the concept of evolution through natural selection [Darwin 1859], whose main idea is that a set of solutions for a problem can be evolved like the population of a generic species in nature. The applications of genetics algorithms are present in many areas, extending from control engineering for non-linear systems identification [Wang et al. 2003] to biomedicine prosthesis development [Wang et al. 2009] and even economy to forecast the behavior of agents [Bharathi et al. 2007].

In this context, a solution is called an individual and was represented by a string of parameters. The first population was instantiated randomly to its full extent since there are no clues concerning how good an initial set of predefined parameters can be. This population passes through a fitness function that indexes a score to each individual, and this function is responsible for assessing how good - or how adapted - the solution that is being evaluated is. After evaluating the population, a group of individuals is chosen as the parents of the next set of solutions, which would compose a new generation; there are countless ways of performing the selection of the parents to the new generation of offspring, and this work gave preference to picking the higher individuals on the scoring system, what is called Elitism [ELI 2006]. Each pair of parents was submitted to crossover in order to generate two offspring solutions, and in the end of the process each offspring might have presented mutation - everything according to predefined rates. The crossover took place in 95% of the reproduction, while the mutation rate assumed the rate of 1% [Grefenstette 1986].

For the Five-State FSM, 22 parameters required adjustment, which originated, in different quantities, from each state separately and also the *transition* function, as follows:

- *Transition Function* has 3 parameters;
- *Approaching Curve* has 4 parameters;
- *Straight Line* has 4 parameters;
- *Out of Track* has 7 parameters;
- *Stuck* has 4 parameters.

However, for the Three-State FSM, only 17 parameters demanded adjustment, which are divided as follows:

- *Inside Track* has 6 parameters;
- *Out of Track* has 7 parameters;
- *Stuck* has 4 parameters.

The source codes for both models presented in this section are available at the *GitHub* repository provided in the references [Git 2015].

# 5 Experimental Results

The main reason of developing the presented controller was to compete in the SCRC. All the taken tests aim to verify the competitiveness of both controllers towards this model of competition. Experiments presented in this section resemble the stages held in the actual championship.

## 5.1 Methodology

Once the models - and which of their parameters required tuning - had been defined, the genetic algorithm was applied to each approach separately, in order to adjust their configurations to obtain superior and competitive status. At the same time, the goal was to find general and versatile controllers with good results for any track and specific ones that were fittest to race in various tracks: road, dirt and oval alike. Therefore, due to the differences observed in the parameters of controllers evolved on dirt and road tracks, the evolution process took place separately for those two kinds of environments, which produced two contrasting sets of enhanced values for each model.

The *metric* chosen to evaluate the generated controllers was the combined sum of the distance raced by the car alone in the first 10 000 game cycles - also called *game tics* - in a list of mixed tracks, this exact metric is used in the qualifying stage of the SCRC. This value will henceforth be called the *fitness* of the controller, as it was used to determine whether he would remain in the evolution process.

The experiments concerning Oval Tracks would repeatedly provide inconclusive results, for example, for the Five State controller the transition function is not triggered because the behavior of the Staright Line is able to solve the whole track. More over in the Three States controller parameters that handle target speed and steer policy are not used. So they were neglected in this evaluation process. Thus, in order to find the best set of parameters for a general track, the two finite state machines were evolved in three different sets of tracks, one with four Dirt Tracks, one with four Road Tracks and another with the four of each type. The evolution process for each set of tracks consisted on 616 generations [Alander 1992] of 30 individuals and culminated in one controller; in other words, at the end of the experiments, there were six evolved controllers, one specific for Road Tracks for each model, one specific for Dirt Tracks for each model, and also one evolved in a mixed manner for each model. Additionally, as the *Stuck State* was only triggered in very specific situations, it was not evolved with the controller and its parameters were hand tuned.

Pursuing an unbiased choice of parameters, the Online Learning module described in Subsection 4.3 was turned off during the evolution progress as it seizes the responsibility for the behaviour of the car for itself during the race and could interfere with natural selection. The validation then occurred through testing the six produced pilots in a predefined set of tracks different from those in which they were evolved, avoiding the evidence of too track-limited parameters. On behalf of comparison, the results from the AUTOPIA controller were incorporated in the analysis, since it can be considered the State of the Art, displaying one of the best performances for the SCR Championship, and should provide satisfactory basis for appraisal.

The four Road Tracks used in the evolution process were chosen from the TORCS standard track set, which are *Spring*, the longest track available on TORCS with more curves than any other track, *Wheel 2*, the most difficult track with sharp and hard curves, *E-Track 3*, a fast track with turns that put to test the dexterity of the controller, and *Forza*, a track considered to be raced fast and whose curve pattern is usually found in others tracks. Also four Dirt Tracks were selected to be used in evolution, which were *Dirt2*, *Mixed1*, *Mixed2*, *Dirt6*. *Dirt2* is a difficult track with close curves, while *Mixed1* is an easy one with few curves, also, *Mixed2* has many turns with medium difficulty, and *Dirt6* present sections where the car drive out the track.

Once the evolution process was finished, the six resultant controllers were tested in the evaluation set of tracks, different from those where they were evolved. Three Road Tracks were picked to evaluate the controllers, they are also available on TORCS and were used both in the competitions of 2008 and to evaluate the *AUTOPIA* controller [Onieva et al. 2009]. *Street-1* and *D-Speedway* were used at the *IEEE World Congress on Computational Intelligence*; and *CG Speedway 1* was used at the *Computational Intelligence and Games Symposium - CIG*. This set also contained three Dirt Tracks, which were *Dirt1*, *Dirt3*, *Dirt4*. *Dirt1* has smooth curves and allows the controller to perform in high speed, while *Dirt3* is an easy one with few curves, also, *Dirt4* has many turns with medium difficulty and is the longest Dirt Track available.

## 5.2 Results

It is important to mention that before each result displayed in that section a warm-up stage was held for 5 laps in each track.

The controllers were submitted to a set of evaluation races and the distances they covered in 10 000 game tics inside each of those races were joined in Table 1. This table also displays how the evolved controllers performed in comparison to the AUTOPIA controller.

In order to have a more in-depth comparison, Table 2 was assembled to display the time elapsed during 10 laps for each of the previous tracks tested. The *Berniw Hist4* bot provided by the TORCS distribution was also used in this test phase even though it uses a different car from those allowed in the SCRC [Loiacono et al. 2010] and has a low performance compared to the others bots provided. AUTOPIA also uses this bot for comparison [Onieva et al. 2012].

It is important to mention that the bots have a full view of the track format and do not use the sensors provided by SCR. Instead, they have directly access to all the information necessary to race, which gives them some advantage over the controllers developed for the TORCS environment, such as the ones described in this paper, which receive only the information that comes from sensors and in addition have to interpret them in order to abstract how the track really is.

## 5.3 Analysis

The best pilots from each race evaluated by Tables 1 and 2 were highlighted in bold. Both sets of experimental results shown at these tables were selected in order to match previous tests performed by the AUTOPIA controller. In 2009, the authors of AUTOPIA [Onieva et al. 2009] evaluated it through racing for a predefined time period and computing the total distance reached, and these results were incorporated in Table 1 for comparison. Differently, in 2012, this controller [Onieva et al. 2012] was assessed by racing alone during 10 laps and calculating the time elapsed to do so, and these results were also incorporated in the evaluation process of this paper and are displayed in Table 2. The tracks that are not present in either of the tests executed by AUTOPIA were validated only between the approaches proposed by this paper, which was informed in Tables 1 and 2 through the "*" symbol, meaning that AUTOPIA had no results for that race in particular.

For a wider analysis not only the numeric results were taken into account in this section but also observations made in the graphic mode provided by the TORCS distribution.

### 5.3.1 Comparing the Controllers Proposed

The overall comparison between the two approaches presented favored the Three-State FSM, on account of the considerably superior results it produced in all the tracks tested. The Five-State FSM presents a very complex transition function that takes into account the variance of the sensorial input to decide if the car is in a straight line, approaching a turn or in a turn. On the other hand, the Three-State FSM has only a single state for handling normal driving situations and it is very easy to say if the controller is inside or outside the track only by checking the track sensor. By having multiple states acting inside the track the Five-State FSM leads to a struggle in defining the boundaries of a curve, the characteristics of an approaching curve and how those situations are different from a straight line. Mismatching emergent from the transiction function would often cause the car to leave the track. It seemed that a single state for handling those situations ended in more accurate behavior on account of having no dependence from an external function.

This contrast in behavior is then interpreted to be the reason of the overwhelming difference in performance, the complexity of the transition function, which supports the initial hypothesis of the evaluation. Due to this attribute, the Five-State FSM undergoes a lot of damage in its car, which can be noted in Table 2 where all the "†" symbols represent individuals that did not finish the race for the reason of reaching the maximum damage permitted.

Because the Three-State FSM demonstrated better results than the other approach proposed, it was elected to be subject of analysis on the evaluation process. As expected, the controller evolved only on Road Tracks was the fastest one. These tracks provide an environment susceptible to high speeds, since its curves are smoother and the friction experienced by the car is higher than the ones from Dirt Tracks. These factors, when combined, allow the controller to race without having to steer too abruptly and to brake without losing control while racing in Road Tracks. Consequently, as the friction increases, steering becomes more accurate in road tracks, practically eliminating critical skidding. Therefore, the result from this end of the evolution process was an aggressive driver with high base-speed.

Dirt Tracks provide a more difficult environment for the pilot to fit in. Sudden braking in tracks of this type often results in unwanted behavior, skidding is noticeably more common then. The driver evolved in this end of the evolution process tends to drive in a low speed so it can keep itself inside the boundaries of the track. Speed driving results in higher damage outcomes and even in the total loss of the car in critical situations. The result obtained was a very careful driver with a low base-speed, and an early brake policy - the car starting to brake far before the turn. This passive driving pattern obtained the smallest distance covered both for the Three-State and the Five-State FSMs.

The driver evolved in a mixed set of tracks combines characteristics from both of them. It drives in a reasonable speed comparing to the first one, but also has the preventive brake policy from the second one. This last end of the evolution process achieved better results than the dirt-evolved behavior in all the tracks tested and outperformed the road-evolved one in every single dirt track. From this information gathered, it was inferred that the controller evolved in mixed tracks tries to reach higher speeds even though this means leaving the track in some turns, mostly because the time spent trying to get back to the racing lane is compensated by the speed of the car. The aggressive behavior inherited from the road-evolved end of the evolution makes this latest controller receive ample damage when leaving the track, and also causes it to hit walls, which resulted in the premature ending of some of the tested races, due to reaching the maximum acceptable damage.

### 5.3.2 Comparing the Three-State FSM to AUTOPIA

Once the Three-State FSM was demonstrated to be more suitable to competitive environments due to its superior performance regarding the Five-State FSM, it was compared to the renowned controller AUTOPIA. Using the distance covered after racing alone in Road Tracks for 10 000 game tics as metric, the Three-State FSMs evolved in Road Tracks and in mixed tracks were able to overcome AUTOPIA in 1 of the 3 tracks tested, as displayed in the bold values in Table 1. The road-evolved Three-State FSM was the controller that got closer to this State of the Art approach using the "distance raced", which comes to endorse the assumption of it being a competitive proposal.

However, while racing alone for 10 laps and computing the time elapsed as metric, AUTOPIA outperformed every controller proposed, just as can be seen in Table 2. Even though the Three-State controller with Road evolved parameters outperformed AUTOPIA in the first 10 000 tics it was not capable of maintaining the advantage in longer races. The road evolved pilot presents a more aggressive behavior even though it means taking more damage it has a gain in performance for the early stages of the race. Although when racing for more than a couple laps the controller becomes more careful after each lap reducing it speed to maintain itself inside the track.

The online learning module plays a crucial role in the overall controller's performance as it prevents unwanted situations to repeat, for example leaving the track. Although this strong dependence might result in performance loss as the controller will gradually reduces it speed after each lap in those points where it leaves the track. More accurate actuators control may reduce the dependence of this module and therefore improves performance.

These results can be used to infer that the Five-State and the Three-State FSMs have a great deal of improvement to achieve when it comes to endurance. The Five-State FSM received total loss and did not complete almost every test performed, ending only one race using this metric. The Three-State FSM, on the other hand, completed practically all the tracks, but did not surpass AUTOPIA in

**Table 1:** *Distance covered in meters racing alone for 10 000 game tics*

| Driver | Street-1 | D-speedway | CG-speedway | Dirt-1 | Dirt-3 | Dirt-4 |
|---|---|---|---|---|---|---|
| FSM3(road) | **7925.6** | 13196.5 | 8745.49 | 3978.01 | 3451.26 | 6757.83 |
| FSM3(dirt) | 2149.77 | 2450.84 | 1951.93 | 3525.84 | 4905.58 | 5590.78 |
| FSM3(mixed) | 7219.28 | 12772.4 | 8126.12 | **4386.64** | **5481.15** | **6939.83** |
| FSM5(road) | 3822.76 | 3427.11 | 4114.66 | 2145.49 | 2205.97 | 3260.19 |
| FSM5(dirt) | 1267.83 | 2936.82 | 4114.66 | 1072.92 | 2205.82 | 3260.33 |
| FSM5(mixed) | 3822.99 | 3427.06 | 4114.8 | 2145.75 | 2205.83 | 3260.31 |
| AUTOPIA | 7091.8 | **15612.3** | **8970.4** | * | * | * |

**Table 2:** *Time elapsed in seconds racing alone for 10 laps*

| Driver | Street-1 | D-speedway | CG-speedway | Dirt-1 | Dirt-3 | Dirt-4 |
|---|---|---|---|---|---|---|
| FSM3(road) | **1086.3** | 607.4 | **495.0** | † | 1150.1 | 1756.4 |
| FSM3(dirt) | † | 840.0 | 1274.8 | 597.3 | 1089.4 | 1307.5 |
| FSM3(mixed) | 1216.50 | **572.60** | 613.77 | 530.0 | 842.9 | 1005.9 |
| FSM5(road) | † | † | 816.6 | † | † | † |
| FSM5(dirt) | † | † | † | † | † | † |
| FSM5(mixed) | † | † | † | † | † | † |
| Berniw Hist4 Bot | 1143.77 | 656.24 | 605.76 | 460.95 | 872.97 | 1127.45 |
| AUTOPIA | * | * | * | **339.3** | **742.4** | **796.5** |

either of them. In order to enhance the endurance feature in the controllers proposed, more robust behavior concerning situations in which the car might crash must be taken into account.

# 6 Conclusions

This paper proposed two approaches developed to control a car during a race in a simulated computational environment, the game platform TORCS. The models of both of these controllers were described, explained, enhanced by means of a genetic algorithm, compared and then tested together with a State of the Art controller - AUTOPIA. It was implied before the testing phase that a finite state machine too burdened in the process of transition between states might lose performance, which was corroborated by the experimental results of the comparison of the two models detailed.

## 6.1 Conclusions

A veil was put over the Five-State FSM from the very beginning of the experimentation phase, which was its great dependency towards the transition function. Early superficial evaluations of the performance of this first model indicated an overcharge concerning this function, which was verified by considerable changes in behaviour derived from adjustments in its parameters. For that reason, a second model with less states - the Three-State FSM - was designed regarding this characteristic and releasing part of the performance burden from the transition function.

The finite state machine with less states achieved a superior overall performance in the tests carried out, in relation to the one with more states. The simplifications fashioned in the transition function of the former were inferred to be the reason for this improvement, along with its intricate relation with the number of parameters that were target of fine tuning in the evaluation and validation process.

The evolution procedure adopted concerning the controllers culminated in three characteristic behaviors. The controller evolved on road tracks became a fast driver, whose hastiness resulted in a careless attitude in general; in other words, it was only good for races in road tracks. The one evolved on dirt tracks turned out to be too careful in contrast, limitedly determining its speeds and, in efficiency terms, inferior. The controller evolved on a mixed set of tracks inherited characteristics from both the previous ones, becoming swift but not too hasty, prudent but not too slow. The latter surpassed the performance of the dirt-evolved drivers even on dirt tracks, and did not lose by far on road tracks in comparison to drivers evolved solely in them.

In terms of speed, the Three-State FSM was able to overcome AUTOPIA in one of the three tracks used to validate the drivers using the distance covered in 10 000 game tics as metric. However, in comparison to the same controller, using the time elapsed to race 10 laps as metric instead, the experimental results provided an insight on the lack of endurance that the finite state machine drivers proposed possess.

To sum up, the interpretation of the global results from the experiments performed gives margin to declare that finite state machines are a reliable technique to implement artificial intelligences, at least for computer games such as simulated races. They provide the possibility of parallel development and also enable parameter tuning in separate fronts, due to the independence and abstraction between the behaviors from each state. Finite state machines also represent a valuable tool for describing an operation model for a process, simplifying and gathering possible situations it might present into straightforward categories of accessible understanding.

## 6.2 Future Works

One characteristic that has been marked as a deficiency in the controllers presented in this paper is the lack of endurance. In order to prevent this from affecting the general performance of the controllers developed, more robust techniques must be integrated into their model. Ways of treating this matter range from harsher brake policies to drive planning intensification, which are already being taken into account for future proposals.

Anti-lock Breaking System (ABS) filter is commonly used in the racing environment [Muoz et al. 2010] and was not implemented in the proposed controller. The Traction Control System (TCS) [Muoz et al. 2010] could also be implemented by applying filters to the acceleration actuator This policies minimize skidding and increase the stability when speed needs to be adjusted in a turn.

The driver uses some parameters that are necessarily greater than others. A model based on fuzzy logic [Perez et al. 2009] could simply solve this ordering issue.

Another important task to be accomplished is the opponent treatment in real-time races. Routines to reduce collisions, i.e., avoiding being overtaken and also being concerned about overtaking the opponents is a fundamental issue. Ignoring adjacent cars usually causes the driver to face unexpected collisions, ending up stuck, considering a worst case scenario. Many of the renowned developers for TORCS already incorporate such treatment in their controllers, and neglecting this necessity renders any driver less robust to unexpected race events, and also reduces its performance.

## References


ALANDER, J. 1992. On optimal population size of genetic algorithms. In *CompEuro '92 . 'Computer Systems and Software Engineering',Proceedings.*, 65–70.

BARTH, M. 2013. The potential role of vehicle automation in reducing traffic-related energy and emissions. *International Conference on Connected Vehicles and Expo (ICCVE)*.

BEJCZY, A. K. 2012. New challenges for application of robotics and automation in space research. *7th IEEE International Symposium on Applied Computational Intelligence and Informatics*.

BERNADEN, A. 1999. Variable air volume control using a finite state machine. *Control Conference (ECC)*.

BHARATHI, R., KUMAR, M. J., SUNITHA, D., AND PREMALATHA, S. 2007. Optimization of combined economic and emission dispatch problem a comparative study. *International Power Engineering Conference*.

BUCKLAND, M. 2005. State-driven agent design. In *Programming Game AI By Example, 2nd ed.*, W. Publishing, Ed. Wordware, Plano, TX, ch. 2, 43–45.

CARBAUGH, J., GODBOLE, D., AND SENGUPTA, R. 1997. Tools for safety analysis of vehicle automation systems. *Proceedings of the American Control Conference*.

CARDAMONE, L., LOIACONO, D., AND LANZI, P. L. 2014. *Simulated Car Racing Championship - Competition Software Manual* - http://arxiv.org/pdf/1304.1672.pdf.

DARWIN, C. R. 1859. On the Origin of Species. In *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.*, J. Murray, Ed.

DE FREITAS, L. G., REFFATTI, L. M., DE SOUSA, I. R., CARDOSO, A. C., CASTANHO, C. D., BONIFÁCIO, R., AND RAMOS, G. N. 2012. Gear2d: An extensible component-based game engine. In *Proceedings of the International Conference on the Foundations of Digital Games*, ACM, New York, NY, USA, FDG '12, 81–88.

DUNCAN, J. C., AND FETERLE, L. C. 2000. The use of personal computer-based aviation training devices to teach aircrew decision making, teamwork, and resource management. In *Proceedings of the IEEE 2000 National Aerospace and Electronics Conference*, 421–426.

2006. *Watchmaker Framework for Evolutionary Computation - Chapter 3. Selection Strategies and Elitism* - http://watchmaker.uncommons.org/manual/ch03s06.html.

2015. GitHub repository for the FSMDriver. Website: https://github.com/bruno147/fsmdriver.

GREFENSTETTE, J. J. 1986. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics SMC-16*, 1 (Jan).

HOLLAND, J. H. 1984. Genetic Algorithms and Adaptation. In *Adaptive Control of Ill-Defined Systems*, vol. 16 of *NATO Conference Series*, 317–333.

HYUNKIM, D., KWON, J., CHEN, K., AND CHOI, K. 2013. Finite state machine for vehicle detection in highway surveillance systems. *Frontiers of Computer Vision, (FCV)*.

LOIACONO, D., LANZI, P. L., TOGELIUS, J., ONIEVA, E., PELTA, D. A., BUTZ, M. V., LÖNNEKER, T. D., CARDAMONE, L., PEREZ, D., SÁEZ, Y., PREUSS, M., AND QUADFLIEG, J. 2010. The 2009 Simulated Car Racing Championship. *IEEE Transactions on Computational Intelligence and AI in Games 2* (June).

MITCHEL, T. M. 1997. Reinforcement learning. In *Machine Learning*, W. Publishing, Ed. McGraw-Hill Science/Engineering/Math, Plano, TX, ch. 13, 385.

MUOZ, J., GUTIERREZ, G., AND SANCHIS, A. 2010. A human-like torcs controller for the simulated car racing championship. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, 473–480.

ONIEVA, E., PELTA, D. A., ALONSO, J., MILANES, V., AND PEREZ, J. 2009. A modular parametric architecture for the torcs racing engine. *IEEE Symposium on Computational Intelligence and Games*, 256–262.

ONIEVA, E., PELTA, D., GODOY, J., MILANÉS, V., AND RASTELLI., J. P. 2012. An Evolutionary Tuned Driving System for Virtual Car Racing Games: The AUTOPIA Driver. *International Journal of Intelligent Systems, Wiley-Blackwell 27*, 3, 217–241.

2014. *7 Reasons to Justify an Operator Training Simulator:* https://www.honeywellprocess.com/library/news-and-events/presentations/Hon-EMEA14-Operator-Training-Simulator.pdf.

PEREZ, D., RECIO, G., SAEZ, Y., AND ISASI, P. 2009. Evolving a fuzzy controller for a car racing competition. *Computational Intelligence and Games (CIG)*.

QUADFLIEG, J., PREUSS, M., AND RUDOLPH, G. 2014. Driving as a human: a track learning based adaptable architecture for a car racing controller. *Genetic Programming and Evolvable Machines - Springer Science+Business New York*.

2013. SCR Championship State of the Art. http://pt.slideshare.net/dloiacono/gecco13scr?related=1.

SHOTTON, J., FITZGIBBON, A., COOK, M., SHARP, T., FINOCCHIO, M., MOORE, R., KIPMAN, A., AND BLAKE, A. 2011. Real-time human pose recognition in parts from single depth images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, CVPR '11, 1297–1304.

2015. *Why use simulation? - Justifying Simulation (accessed: 2015):* https://www.promodel.com/pdf/Justifying%20Simulation.pdf.

TERZIC, I., ZOITL, A., AND FAVRE, B. 2008. A survey of distributed intelligence in automation in european industry, research and market. *IEEE International Conference on Emerging Technologies and Factory Automation*.

WANG, Q., SPRONCK, P., AND TRACHT, R. 2003. An overview of genetic algorithms applied to control engineering problems. In *International Conference on Machine Learning and Cybernetics*, vol. 3, 1651–1656.

WANG, S., HONG, J., LIU, Z., AND WANG, W. 2009. Method of human-imitating posture generation for mechanical hand prosthesis. *ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots*.

WHITTON, N., AND MOSELEY, A. 2012, ch. 8. Authentic contextual games for learning. In *Using Games to Enhance Learning and Teaching: A Beginner's Guide, 2nd ed.*, N. Whitton and A. Moseley, Eds. Routledge, New York, NY.

WYMANN, B., ESPIÉ, E., GUIONNEAU, C., DIMITRAKAKIS, C., COULOM, R., AND SUMNER, A., 2014. TORCS, The Open Racing Car Simulator. http://www.torcs.org.

WYMANN, B., DIMITRAKAKIS, C., SUMNER, A., ESPIE, E., AND GUIONNEAU, C. 2015. *TORCS: The open racing car simulator (accessed: 2015):* `http://www.cse.chalmers.se/~chrdimi/papers/torcs.pdf`.

XUE, H., WU, H., MENG, X., AND LIU, W. 2011. Simulation research and programming of torque fluctuation of automotive steering system. In *Second International Conference on Mechanic Automation and Control Engineering*, 4385–4387.

ZHANG, F., DU, Z., SUN, L., AND JIA, Z. 2007. A new novel virtual simulation system for robot-assisted orthopedic surgery. *IEEE International Conference on Robotics and Biomimetics*.