Intelligent And Mobile Robotics Simulated Car Racing with TORCS

Diogo Silva (dbtds@ua.pt)
DETI, University of Aveiro
Aveiro, Portugal

***Resumo* – Este artigo descreve um agente desenvolvido para Robótica Móvel e Inteligente na Universidade de Aveiro. O objetivo principal do robô é controlar um carro de corrida numa pista desconhecida sendo que o tempo com que faz a mesma é o fator principal que se pretende minimizar. Foi considerado um cenário onde o agente corre sozinho na pista. Este artigo cobre fatores como seleção da mudança adequada, pressão no travão ou acelerador e a direção do guiador. Para além disso também é abordado como foi feita a escolha do caminho que o agente deve percorrer de forma a minimizar o tempo da corrida.**

***Abstract* – This paper describes an agent developed for Intelligent and Mobile Robotics at the University of Aveiro. The main objective of the robot is to control a race car in an unknown track. The agent will try to minimize the time that it takes to complete the race. It was considered that the agent always runs alone in the track. This article covers several factors like gear selection, pressure on the accelerator or the brakes and direction of the steer. Besides that, it is also taken into account the path that the agent takes to minimize the time to complete the race.**

***Keywords* – robotics, agent, ua, rmi, torcs, fuzzy, vfh, race, car, mapping**

I. INTRODUCTION

This project was created for the course unit Intelligent and Mobile Robotics at the University of Aveiro. It consists in developing an autonomous agent, which is a system that tries to understand the environment using sensors and acts according with it using actuators. Two approaches were taken into account, between a deliberative agent (where it senses, plans and then actuates) or a reactive agent (where it senses and then actuates; there is no planning).

This agent will have to find its way to the final target through the track. There is only one possible path, although several aspects might be taken into consideration: the robot might want to be close to the edge, or not, depending on the situation; The track only contains curves, some of them straight forward, other more tight and harder to pass by; The agent uses several sensors to sense what is going on in track and several actuators to completely control the car.

Besides the description of the problem, the strategies adopted to fulfil every challenge are described, e.g. the strategy adopted by the agent to obtain the maximum speed during the path without leaving the track.

Every aspect of this paper was simulated using a specific platform and was never implemented in a non-simulated environment.

II. SCENARIO

A. Description

The scenario is simple; it contains the road, where the car usually has some good friction, and outside of the road, where it usually has less friction than in the road, like sand. It is possible to verify that scenario in the image 1.

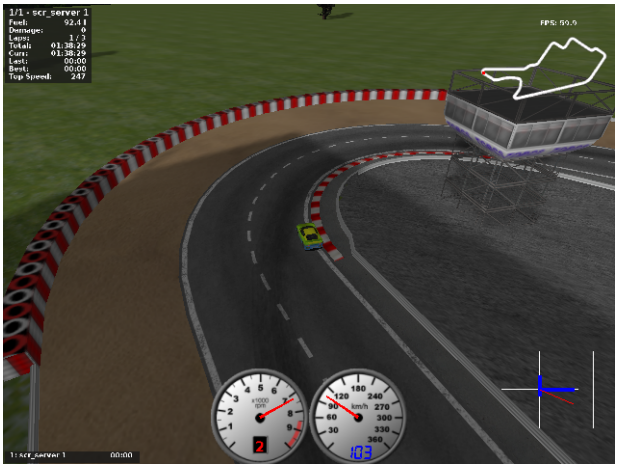

Figure 1
CAR RUNNING THE TRACK USING TORCS

In the simulation, it is possible to visualize the track where the car is moving, on the top right corner. The agent does not have any kind of sensors that allow to visualize the full track, it only allows to visualize 200 meters around the car with several laser range finder sensors.

B. Platform

The scenario is simulated using a platform known as TORCS, The Open Racing Car Simulator, available at `http://torcs.sourceforge.net/`. Moreover, the version used is given with the agent's source code.

TORCS allows the development of agents directly on the platform and get compiled with it, although there is a patch [1] that was developed for a Championship related with TORCS. That patch uses sockets to communicate with TORCS instead of compiling with the simulator, like shown on the following image [2] 2.

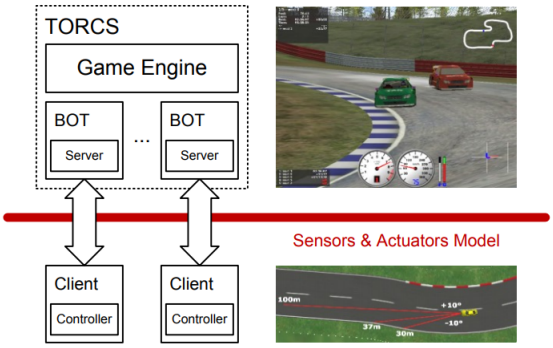

Figure 2
COMMUNICATION USED BETWEEN TORCS AND AGENTS AFTER THE PATCH [2]

That patch does not only applies sockets, it also restricts some sensors and some information. In a normal agent developed within TORCS it is possible to access the full track and the global angle of the robot, which affects a lot. After applying the patch, those information are no longer available.

III. DESCRIPTION OF THE ROBOT'S SENSORS AND ACTUATORS

The robot has several sensors and actuators associated to it, that allow to take completely control of the car on the track. There is a list of the sensors and the actuators within the source code with the name 'sensors_description.md'.

A. Sensors

The sensors that are available after the patch being applied to the simulator are the following:

- angle [-π ,+π] (rad) - Angle between the car direction and the direction of the track axis;
- gear -1,0,1,2,3,4,5,6 - Current gear: -1 is reverse, 0 is neutral and the gear from 1 to 6;
- speedX (-∞,+∞) (km/h) - Speed of the car along the longitudinal axis of the car;
- speedY (-∞,+∞) (km/h) - Speed of the car along the transverse axis of the car;
- speedZ (-∞,+∞) (km/h) - Speed of the car along the Z axis of the car;
- track [0,200] (m) - Vector of 19 range finder sensors: each sensors returns the distance between the track edge and the car within a range of 200 meters. When the car is outside of the track (i.e., trackPos is less than -1 or greater than 1), the returned values are not reliable (typically -1 is returned);
- trackPos (-∞,+∞) - Distance between the car and the track axis. The value is normalized w.r.t to the track width: it is 0 when car is on the axis, -1 when the car is on the right edge of the track and +1 when it is on the left edge of the car. Values greater than 1 or smaller than -1 mean that the car is outside of the track, as shown on image 3;

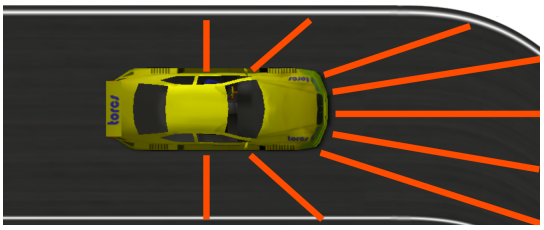

Figure 3
TRACKPOS VISUALIZATION EXAMPLE (GECCO PRESENTATION [3])

- wheelSpinVel [0,+∞] (rad/s) - Vector of 4 sensors representing the rotation speed of wheels.

There are more sensors that the ones described in here, although the sensors in the list were the ones used. The sensors that were not used are: curLapTime; distRaced; distFromStart; opponents; z; focus; fuel; lastLapTime; racePos.
Some of those sensors were only being printed to compare with other racers, like distRaced since some comparisons appear in terms of distance raced by the car.

B. Actuators

The actuators that are available after the patch being applied to the simulator are the following:

- accel [0,1] - Virtual gas pedal (0 means no gas, 1 full gas);
- brake [0,1] - Virtual brake pedal (0 means no brake, 1 full brake);
- clutch [0,1] - Virtual clutch pedal (0 means no clutch, 1 full clutch);
- gear -1,0,1,2,3,4,5,6 - Gear value;
- steering [-1,1] Steering value: -1 and +1 means respectively full right and left, that corresponds to an angle of 0.366519 rad;
- focus [-90,90] Focus direction in degrees;
- meta 0,1 This is meta-control command: 0 do nothing, 1 ask competition server to restart the race.

IV. STRATEGY OVERVIEW

It was taken into consideration two different approaches:

- Deliberative agent - In a first phase, the agent moves with reduced velocity to prevent to slip on the track. During that phase, the agent will apply kinematics to discover its position and create a mapping according with it, based on the sensors received by trackPos.
  After the agent gets the mapping done (after the first lap), it will go to a normal speed, although it cannot use kinematics otherwise it will lose its position. The idea was to apply Monte Carlo after the first lap taking into account the current map.

After knowing the position, applying a minimum curvature path (MCP) was desired, but the mapping was done applying a grid strategy, which makes it hard to get a path based on the minimum curvature.
The ideal scenario was to get the mapping done with segments. Since the mapping done was not perfect for applying MCP, a second approach was taken into account.

- Reactive agent - This second approach is simple: gear and clutch are controlled manually with basic functions (section X), steering is controlled by switching between a vector field histogram and a middle track approach (section IX) and brakes/gas pedal are controlled using a Fuzzy Controller (section XI).

V. MAPPING

To perform the mapping, the agent uses the obstacle sensors and the ground sensor. Mapping does not take into account error generated by the movement model, it just takes into account the error from the sensors.

A. Estimation of the robot's pose

Since the robot uses its position when exploring to know if it is in an area already visited, estimation of the robot's position was one of the first things that has been done.

To calculate the robot's pose, it needs to know how much each wheel has moved. The robot knows the speed of each wheel, so what it needs to do is to convert that speed into a distance, knowing that the time between two measures is the TORCS simulation step (20 ms).

Now that the agent knows the distance that each wheel has moved (out right and out left), it can calculate its position using the following formula for its movement:

$$lin = \frac{out_{\text{right}} + out_{\text{left}}}{2}$$

$$x_t = x_{t-1} + lin * \cos(\theta_{t-1}) \quad y_t = y_{t-1} + lin * \sin(\theta_{t-1})$$

It estimates the new global angle using the following formula:

$$rot = \frac{out_{\text{right}} - out_{\text{left}}}{robotDiam}$$

$$\theta_t = \theta_{t-1} + rot$$

Where 'robotDiam' is the distance between the front wheels and the 'out' variables (right and left) are the distance of each wheel.

B. Structure to store the mapping

The structure used is a bi-dimensional vector which contains a wall counter, ground counter, visited flag and a state (might be SAFE, UNSAFE or UNKNOWN).

Every time that a counter is incremented, the state is recalculated and it is the only value that is available for anyone that uses the Map; other values are internal.

The result obtained from it can be visualized in the following image 4.

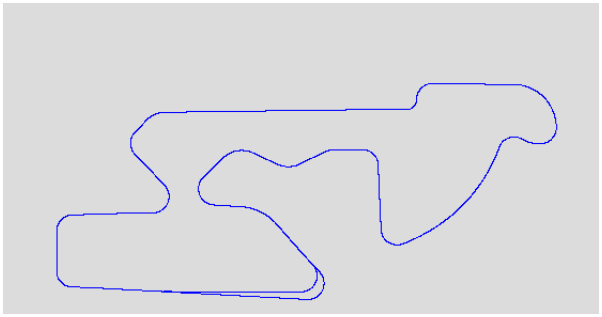

Figure 4
MAPPING USING THE KINEMATICS CALCULATED

It is possible to visualize some error in the bottom right curve; it might be some slippage that has occurred.

VI. BEST RACE PATH WITH MAPPING

Before introducing the laser rangefinders into the mapping, some investigation was done related with the best path for the race.

A. Best Racing Path vs Shortest Path

The shortest path seems to be a good option that is considered most of the time, although when talking about race paths, that is not an option. E.g. if there is a tight curve on the left, the car should start from the right side of the road instead of the left in order to prevent a lose of speed.

There is a strategy that takes that into consideration, which is minimum curvature path. It always tries to calculate a path where the curve is minimum. Most of the time in shortest path the curve will be the worst one.
In the following image 5 it is possible to visualize the difference.

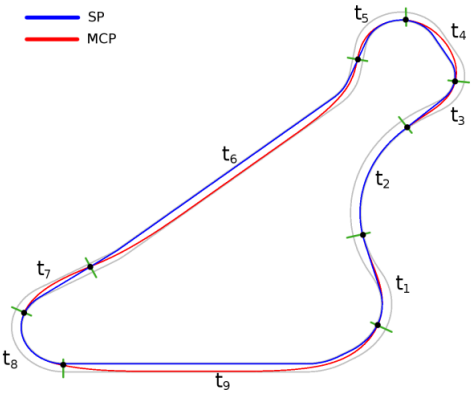

Figure 5
MINIMUM CURVATURE PATH VERSUS SHORTEST PATH [4]

B. Minimum Curvature Path calculation

A problem was found while trying to calculate these curves: they are easier to calculate when the mapping is done based in segments of road instead of grid, which makes it impossible to use with the current mapping. Therefore a different approach was taking into account.

VII. Deliberative approach conclusion

The deliberative approach for the robot was left behind because there was some error related with the kinematics of the agent and in dirt tracks it could easily get lost due to the friction of the road.
The other reason was related with the calculation of the minimum curvature path, which was a hard task to do using a grid mapping. The mapping had to be done with segments instead of grid.

After those conclusions, an approach based in a reactive agent was taken into account.

VIII. Reactive Racing Path

For a reactive agent, which does not plan, there are two different possibilities to drive on the road that the agent takes into consideration.

A. Staying in the mid of the track

The basic strategy is to stay in the middle of the track. That was done using directly the sensor trackPos. That sensor returns the position of the car in the track, where 0 means that it is in the middle. So all that the car had to do was to focus into guarantee that the trackPos was always close to 0.

There is an evident negative aspect about this: it does not cut the curves, which will increase the time a lot and make the curves harder to control for the agent.

B. Vector Field Histogram

A vector field histogram represents a polar visualization of the field; in this case, the road. The agent uses the laser rangefinders to obtain a visualization of his surroundings and then plots a vector field histogram in order to analyze it, like the image 6.

Figure 6
VISUALIZATION OF THE VECTOR FIELD HISTOGRAM

In the previous image it is possible to see that the laser rangefinder reports higher values at the right than on the left, so the agent knows that there is a gap on the right side and it can use it to move alongside with the road.

The advantage of using this method is that it will always cut through the higher distance value and those values always occur when cutting the curve, forcing the car to use the inside part of the curve.

This method has also a downside: when following the road without any noticeable difference in the histogram, the car will just keep its position on the track and it might be too close to the edge.

C. Path final decision

The final decision will take both paths into consideration:

- when the distance of the middle sensor is too high it will drive through the middle of the track;
- when that distance gets closer, it will start moving to the inside part of the curve using the vector field histogram.

.
That behaviour is not perfect, although when comparing with the minimum curvature path, it gets close to it.

IX. Steering Control

The steering control takes into consideration the current angle of the agent in the track, so it can move in the same direction of the road. To apply the respective track position, it is directly subtracted to the current angle, causing a similar effect of a proportional controller.

The other important factor to take into consideration was the speed of the car. If it is moving too fast, the steering wheel will require less movement; when the speed gets lower, it will require an higher movement on the steering wheel.

To achieve that effect, there were taken several points that would be the perfect sensibility for that speed: (0.0, 3); (50, 2); (80, 1); (100, 0.7); (150, 0.5); (200; 0.2), where the first element is the current speed of the car and the second is the movement required on the steering wheel.

After getting those points, they were interpolated using a Lagrange Polynomial, obtaining the following expression:

$f(x) = 2.68253\times10^{-10}x^5 - 1.5146\times10^{-7}x^4 + 0.00003053x^3 - 0.002493x^2 + 0.0455714x + 3$
Where x is the current speed of the car and $f(x)$ is the output, which is the movement required in the wheel.

X. Gear Control

The gear control is defined with a static function where it receives the current rotations per seconds and the current gear, and outputs the gear for the next cycle. If it goes over/under a given threshold it will switch the gear.

That threshold was calculated based on the torque power and not in the rotations per second. The main idea is to keep the torque as high as possible and when it starts to decay, switch the gear.

The following image 7 represents the growth of the torque with the rotations per minute.

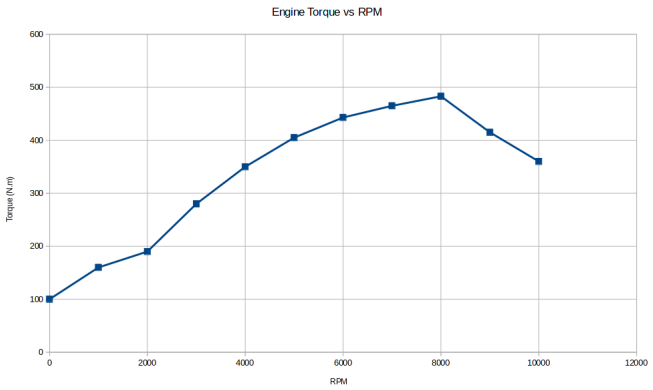

Figure 7
GROWTH OF THE TORQUE WITH THE RPM WITHOUT ANY GEAR RATIO APPLIED

The red line of this car occurs when the rotations reach 9.1k. When there is no gear applied it is possible to verify that the torque is higher at 8k than at 9k and there must be a preference for the peak torque when possible. However, when increasing the gear, it will reduce the torque.

Gear	Ratio	Up	Up RPM
1	3	1.578947368	12631.57895
2	1.9	1.357142857	10857.14286
3	1.4	1.272727273	10181.81818
4	1.1	1.222222222	9777.777778
5	0.9	1.168831169	9350.649351
6	0.77	----------	----------

Figure 8
BEST ROTATIONS TO CHANGE GEAR (NOT TAKING INTO ACCOUNT THE RED LINE)

In the table shown before, it is possible to verify that the best RPM to change the gear is always over the red line. Thus, the car will always change the gear at the red line to prevent damage to the motor.

XI. FUZZY LOGIC CONTROLLER

"Fuzzy logic is a form of many-valued logic that deals with approximate, rather than fixed and exact reasoning. Compared to traditional binary logic (where variables may take on true or false values), fuzzy logic variables may have a truth value that ranges in degree between 0 and 1." [5]

There is 2 actuators that still need to be controlled, the acceleration and the brakes. Those 2 will be controlled using a fuzzy logic controlled.

A. Library used

A library was used to represent the expected behaviour of a fuzzy logic: `http://www.fuzzylite.com/`

This library allows to represent all the logic in a separate file with extension .fcl where it requires to describe the inputs and outputs with the respective ranges and their rules.

B. Inputs

There are 3 inputs into the system:

- maxAngle - Angle between the car and his target objective;
- distFront - Distance between the car and the limits of the track right in front of him;
- speedX - Current car speed;

All the inputs must be mapped into variables to be described by the rules later:

- maxAngle might be MID, CLOSE_MID or FAR - It means that if it is MID, the car is in the right direction, CLOSE_MID means that it is close to the right direction but still has to steer a bit and FAR means that it will have to steer a lot so it might have to brake;
- distFront might be CLOSE (yellow), MEDIUM (orange) or FAR (red) - It is possible to verify that medium is not exactly in medium and close is not exactly the same size as far, that is because the car is extra careful when getting in close ranges to the track limits;

Figure 9
DESCRIPTION OF THE INPUT DISTFRONT - IT RANGES FROM 0 TO 200)

- speedX might be TOOSLOW (red), SLOW (yellow) or FAST (orange)

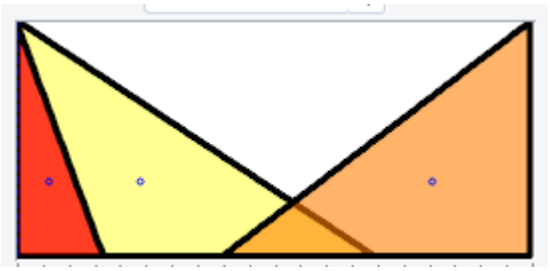

Figure 10
DESCRIPTION OF THE INPUT SPEEDX - IT RANGES FROM 0 TO 300)

C. Output - Acceleration and Brakes

The acceleration and the brakes were represented as the previous variables, using areas where those status could exists.

- Acceleration - It may be NOGAS, NORMAL or FULLGAS;
- Brakes - It may be NONACTIVE, MIDBRAKE or FULLBRAKE.

After the definition of the output, several rules were defined in order to fulfill the problem:

```
if speedX is SLOW then brake is NONACTIVE
if speedX is FAST and distFront is CLOSE then
    brake is FULLBRAKE
if distFront is FAR then accel is FULLGAS
if distFront is CLOSE or distFront is MEDIUM
    then accel is NORMAL
if (maxAngle is CLOSE\_MID.A or maxAngle is
    CLOSE\_MID.B) and speedX is FAST then brake
    is MIDBRAKE
if (maxAngle is FAR.A or maxAngle is FAR.B) and
    (distFront is CLOSE or distFront is MEDIUM)
    then brake is FULLBRAKE
if distFront is CLOSE then brake is FULLBRAKE
if distFront is MEDIUM and speedX is FAST then
    brake is FULLBRAKE and accel is NORMAL
```

The thresholds of the variables were adjusted with the performance that was visualized. No optimization on the threshold has been done, like evolutionary learning.

XII. UNSTUCK ROUTINE

There is a routine responsible to detect when the car in stucked, its important to prevent such situations in case of them to happen. The routine is activated when the car is facing the wrong direction and not moving for more than second and half.

After the car enters in this routine, it switches the gear to reverse and starts positioning the car in the middle of the road again. After half a second of the car being correctly positioned, the car goes back to normal state and starts racing again.

XIII. INITIAL CLUTCH - DRAG

Its important to get an higher initial speed to prevent to take a long time to start up. To reach that objective there is a small implementation that takes control of the clutch in the initial seconds, keeping it higher till the car gets some speed.

To allow the car to get the maximum initial speed as possible, the gear is maintained in 1 and acceleration is always on top.

XIV. AGENT PERFORMANCE AND COMPARISON

In order to compare the other robot, it was taken into account that this agent does not race against other cars. So the best way to compare is to look at the qualification phase of the competition in previous years.

The qualification phase is measured based on the distance raced by the car during 10000 steps of the simulator.

To compare it was used the results from year 2009 in the GECCO LEG and from CIG LEG [6].

Table I
SKYLAKE PERFORMANCE AGAISNT OTHER AGENTS (PART 1)

Driver \Track	Forza	Dirt 3	Alpine 2
Butz & Lonneker	11768.3	6216.2	7798
Onieva & Pelta	11997.2	6659.8	6826.4
Cardamone	9028.3	5439.8	6398.5
Perez & Saez	9028.3	5190.4	6272
Bernardi et al.	6685.9	4993.9	
Vrajitoru & Guse	7176.7	5788.3	
Wong	9781.6	5106.5	
Munoz	10577.6	4459.1	
Chiu	10424.9	901.6	5527.7
Szymaniak	6856.8	782.8	
Quadflieg & Preuss	12191.5	5502	6398.6
SimpleDriver C++	6122.7	3267.9	3216.9
Skylake	**9368.56**	**6382.11**	**6504.37**
Skylake Position	**7**	**2**	**3**

Table II
SKYLAKE PERFORMANCE AGAISNT OTHER AGENTS (PART 2)

Driver \Track	Alpine 1	E-Road
Butz & Lonneker	8677.1	10253.3
Onieva & Pelta	8386.5	9301.1
Cardamone	7308	7716.3
Perez & Saez	7661	6586.6
Bernardi et al.	7184.2	8036.8
Vrajitoru & Guse	6410.3	6384.4
Wong	6480	6638.3
Munoz	5548.7	7573.5
Chiu	6539	5124.3
Szymaniak	3711.9	4501.7
Quadflieg & Preuss	7482.2	
SimpleDriver C++	3376.7	3999.7
Skylake	**6814.2**	**7369.66**
Skylake Position	**6**	**6**

It is possible to verify that Skylake compared with SimpleDriver which was the code based used for the robot is much better, it has almost everytime two times the score of the Simple Driver.

When comparing the agent to other agents implemented that normally participate in the competition, it behaves really well, even in one of the tracks it has ended in second.

XV. BRIEF DESCRIPTION OF THE SOURCE CODE

The source code contains a CMakeLists.txt that is responsible to compile all the code and create the proper linkage with the necessary libraries. Besides the CMakeLists.txt, the source code contains 4 folders:

- core-common - This folder contains the implementation responsible for all the communication with the TORCS and store that data, this should be equal to all agents;
- core - This is the agent Skylake core, containing several constants associated with the car itself;
- mapping-driver - This is the deliberative agent that represents a map using SDL2;
- vfh-fuzzy-driver - This is the reactive agent using VFH and Fuzzy Controller to drive.

There is also a driver used for the Fuzzy Controller inside the folder vfh-fuzzy-driver with the name 'driver_modified.fcl'.

There is a file with the name 'car_description.xml' which contains the description of the car used during the simulation, like distance between wheels, tire size, etc.

There is also a file with the sensors description which has the same content of the manual but simplified.

XVI. CONCLUSION

The agent is considered robust to solve the challenge.

The agent manages to solve any track, ones with more difficulty than others. Some improvements should be taken into consideration:

- Implementation of the Minimum Curvature Path (MCP) using the respective mapping in segments;
- Using a genetic algorithm to optimize the output of the Fuzzy Controller in order to achive the maximum performance.

The agent was tested in several different tracks with different frictions and it managed to solve all of them, ones slower than others but always made it to the end, which is also important.
The car chosen was always the same car1-trb1, which might not be the best one since there is more 7 cars that the programmer could have picked.

REFERENCES

[1] Mohommad Bonyadi, Samadhi Nallaperuma, Daniele Loiacono, Frank Neumann *Simulated Car Racing Championship 2015*. University of Adelaide and the Politecnico de Milano.

[2] Loiacono, Daniele, Luigi Cardamone, and Pier Luca Lanzi. *Simulated car racing championship: Competition software manual.*. arXiv preprint arXiv:1304.1672 (2013).

[3] Daniele Loiacono and Pier Luca Lanzi *2013 Simulated Car Racing*. GECCO-2013

[4] Cardamone, Luigi, et al. *Searching for the optimal racing line using genetic algorithms, Computational Intelligence and Games (CIG)*, 2010 IEEE Symposium on. IEEE, 2010.

[5] Novák, V., Perfilieva, I. and Močkoř, J. *Mathematical principles of fuzzy logic*. Dodrecht: Kluwer Academic. (1999)

[6] Loiacono, Daniele, et al. *The 2009 simulated car racing championship.* IEEE Transactions on Computational Intelligence and AI in Games 2.2 (2010): 131-147.