

Master's Thesis
Computer Science
Thesis no: MCS-2013:19
December 2013



Optimization Techniques For an Artificial Potential Fields Racing Car Controller

A Particle Swarm Optimization approach

Nader Abdelrasoul

School of Computing
Blekinge institute of Technology
SE – 371 79 Karlskrona

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:

Author(s):

Nader Abdelrasoul

Address: Karlskrona, Sweden

E-mail: nader.jo@gmail.com

University advisor(s):

Dr. Stefan Johansson

School of Computing/Blekinge Institute of Technology

School of Computing

Blekinge Institute of Technology

SE – 371 79 Karlskrona

Sweden

Table of Contents

Abstract.....	4
1 Introduction.....	1
1.1 Scope and Context	1
1.2 Problem Definition.....	1
1.3 Aims and Objectives.....	2
1.4 Research Questions.....	2
1.5 Approach	2
1.6 Contribution.....	3
1.7 Thesis Outline	3
2 Background.....	4
2.2 Autonomous Vehicle Controllers.....	4
2.3 Artificial Potential Fields (APF).....	5
2.4 Particle Swarm Optimization (PSO).....	5
2.5 TORCS Environment.....	6
2.6 Related Work.....	9
2.6.1 Robot and Vehicle Controllers.....	9
2.6.2 Controllers in Games.....	10
3 Research Methodology.....	11
3.1 Research Design.....	11
4 Modeling the Controller.....	12
4.1 Proposed Method.....	12
4.2 Controller Design.....	12
4.3 The Multi Agent System	15
5 Experiments.....	19
5.1 Variables	19
5.2 Tracks.....	19
5.3 Opponents.....	22
6 Results and Analysis.....	23
6.1 Results.....	23
6.1.1 Experiment 1.....	23
6.1.2 Experiment 2.....	27
6.2 Controller Evaluation.....	30

6.2.1 Experiment 1.....	30
6.2.2 Experiment 2.....	31
6.3 Discussion.....	32
6.3.1 Experiment 1.....	32
6.3.2 Experiment 2.....	33
6.4 Conclusions.....	34
6.5 Validity Threats.....	35
6.6 Future Work.....	35
References.....	36

ABSTRACT

Context. Building autonomous racing car controllers is a growing field of computer science which has been receiving great attention lately. An approach named Artificial Potential Fields (APF) is used widely as a path finding and obstacle avoidance approach in robotics and vehicle motion controlling systems. The use of APF results in a collision free path, it can also be used to achieve other goals such as overtaking and maneuverability.

Objectives. The aim of this thesis is to build an autonomous racing car controller that can achieve good performance in terms of speed, time, and damage level. To fulfill our aim we need to achieve optimality in the controller choices because racing requires the highest possible performance. Also, we need to build the controller using algorithms that does not result in high computational overhead.

Methods. We used Particle Swarm Optimization (PSO) in combination with APF to achieve optimal car controlling. The Open Racing Car Simulator (TORCS) was used as a testbed for the proposed controller, we have conducted two experiments with different configuration each time to test the performance of our APF-PSO controller.

Results. The obtained results showed that using the APF-PSO controller resulted in good performance compared to top performing controllers. Also, the results showed that the use of PSO proved to enhance the performance compared to using APF only. High performance has been proven in the solo driving and in racing competitions, with the exception of an increased level of damage, however, the level of damage was not very high and did not result in a controller shut down.

Conclusions. Based on the obtained results we have concluded that the use of PSO with APF results in high performance while taking low computational cost.

Keywords: Autonomous car controller, Artificial Potential Fields (APF), Particle Swarm Optimization (PSO), TORCS.

1 INTRODUCTION

1.1 Scope and Context

Car racing games are considered exciting to many people. Racing between different competitors makes each one of them eager to achieve better results and win the race to prove superiority in skills and tactics over other competitors. In a car racing game there is usually a human controlling the car during the race, however, it is possible to design an autonomous car controller that can substitute the human driver in carrying out the driving task.

Autonomous racing car controllers are software programs designed to read sensor data from the racing environment as an input. The sensor input is processed by the controller according to a pre-defined model, the result is a set of movement instructions that are used to move the car during the race towards the final destination.

There has been many attempts by researchers to build autonomous car racing controllers using Artificial Neural Networks [1], and genetic algorithms [2]. These approaches have some shortcomings including low performance, high computational expense, and the lack of online reactive behavior, therefore there is a need to introduce new ways for building racing controllers that are computationally inexpensive, and at the same time can achieve good performance and react to new circumstances during races.

An algorithm called Artificial Potential Fields (APF) is used for path finding and collision avoidance in robotics. APF has been used in building a racing controller “Powaah”, which participated in the CIG 2011 and scored 6th out of 10 competitors [3]. The advantage of APF is its simplicity and straight forward approach along with its high efficiency and good results [4].

The draw back of using APF is the lack of optimality in the racing line. An optimal racing line is the one which reduces the time needed to complete the race [2]. APF generates a racing line but does not insure that the racing line is optimal, therefore there is a need to have an optimization technique that will fix this issue.

After studying different optimization techniques, we found that a method called Particle Swarm Optimization (PSO) shows a great suitability to be used with APF. PSO is a highly dynamic optimization approach, it uses a number of moving entities to search a solution space in a manner that is inspired by the way birds search for food [5]. PSO is relatively new, however, it received a lot of attention and has been applied in many practical applications.

1.2 Problem Definition

Building an autonomous car controller is a difficult and challenging task due to the complexity of decision making in the racing environment. Realistic racing environments are highly dynamic, which means that there are many variables constantly changing during the race, the controller has to successfully deal with the input and overcome the complexities associated with making the right decision in order to achieve optimal performance.

The controller must be able to achieve a number of tasks in order to win the race, these tasks includes car movement, maneuverability, and maintaining top speed. Therefore we need to build a model that can calculate and update the car position in a way that fulfills the racing requirements and does not include high computational overhead.

1.3 Aims and Objectives

The main aim is to build an autonomous racing car controller that can achieve high performance. In order to achieve our aim we have a number of objectives that we need to fulfill to reach our target, these objectives are:

- Identify a path finding method for controlling the car movement.
- Identify a suitable optimization method for finding the optimal racing line.
- Propose a general racing car controller structure.
- Implementing the controller using a realistic racing environment.

1.4 Research Questions

As we have stated earlier, we want to build a controller using APF that is capable of reaching high performance, therefore we have added the use of PSO to achieve optimality. Furthermore, since performance is a general term we need to be more specific regarding which aspects of performance are we going to focus on, therefore we have three subquestions that specify the general question. Our research questions are:

RQ 1: Does using PSO enhance the performance of an APF based autonomous car racing controller?

RQ 1.1: Does using PSO enhance the racing speed of an APF based autonomous racing car controller?

RQ 1.2: Does using PSO reduce the total racing time of an APF based autonomous racing car controller?

RQ 1.3: Does using PSO reduce the total damage of an APF based autonomous racing car controller?

1.5 Approach

We have chosen APF and PSO in building an autonomous car controller since the two approaches complement each other, APF in path finding and PSO in finding the optimal racing line. We have tested our implementation using a testbed called The Open Racing Car Simulator (TORCS). TORCS is a platform used by researchers to evaluate new autonomous motion controllers in a simulated racing car environment [6]. Multiple controllers compete against each other which provides a unique environment to assess and compare different algorithms.

Our scope in this thesis was limited to the design and implementation of a racing car controller using the APF-PSO algorithm. We used TORCS as a simulation tool to conduct two experiments for evaluating the proposed algorithm against using pure APF in the first experiment, and against an APF controller and other controllers in the second experiment.

The results of the experiments showed that using the proposed APF-PSO algorithm achieved good results compared to APF. However, the main issue was in the level of damage, since our controller maintained a very high speed throughout different races, which caused us to be in the top three most damaged in a total of six controllers.

1.6 Contribution

This study contributes to the improvement of building autonomous racing car controllers by implementing a well known path finding approach (i.e. APF) with a relatively new optimization technique (i.e. PSO), which resulted in a powerful model with high flexibility while at the same time maintaining the simplicity of the two approaches.

1.7 Thesis Outline

The remaining of this report is organized as follows:

- Chapter 2** Background
In this chapter we present the background of autonomous controllers, APF, PSO, and TORCS environment. We shed a light on the existing work in the area.
- Chapter 3** Research Methodology
In this chapter we present the research design.
- Chapter 4** Modeling the Controller
In this chapter we describe the Multi Agent System (MAS) and the mathematical bases of the controller.
- Chapter 5** Experiment
In this chapter we describe the details of the experiment, dependent and independent variables, tracks, and opponents.
- Chapter 6** Results and Analysis
In this chapter we discuss the results and make conclusions in relation to the research questions. Finally we have threats to validity and future work.

2 BACKGROUND

2.2 Autonomous Vehicle Controllers

Autonomous vehicle motion controlling is a widely growing branch of Artificial Intelligence research. This type of autonomous controllers has been successfully applied in robotics, and in computer games [7] .

Autonomous controllers depend on sensor data obtained from the surrounding environment, the data is processed by the controller, and in return the controller gives the corresponding movement command to the vehicle. Using autonomous controllers has proven to improve the path finding and reduce the risk of collisions and lane departures [8] [9] .

Designing an autonomous controller is a challenging task, since the driving task is subject to many variables within the surrounding environment. In simulated and game environments, the complexity of a controller depends on how realistic is the game environment. A realistic game includes a large number of variables that exists in the real world, such as aerodynamics and sensor noise. This creates a high level of details that must be considered when designing the controller.

As a result of operating a controller that deals with high level of details in a realistic environment performance requirements such as response time becomes harder to meet. This brings up the need for efficient algorithms which can be executed within the time limits and gives good results in performing the controller's tasks such as path finding and collision avoidance.

To build a controller for simulated car racing, we need to identify the main requirements that the controller must be able to fulfill in order to achieve the main goal which is winning a race. The first requirement is that the controller should be able to move the vehicle through altering a set of control variables that sets the vehicle to move through a specified racing track. These variables may include; acceleration, steering, breaking, etc. This requirement can be achieved through passing the corresponding values to the simulation server and receiving sensor input from it. Through this communication with the server, the controller should be able to generate and correct its position.

The second requirement is that the controller should be able to carry out some maneuverability tasks that will give it advantage during the race. Maneuverability tasks include overtaking and avoiding collisions, it can be achieved by using a planning approach that ensures a collision free path by avoiding and overtaking obstacles. The third and final requirement is keeping the highest possible speed during the race, which can be done by following the longest line of sight on the track.

Since winning the race is the main goal, the controller must be able to find the best possible moves by evaluating different alternatives before choosing the next move. To evaluate a move the controller must see the possible gain and loss involved with it, a possible gain can be higher speed or better position in the game i.e. overtaking, and a possible loss can be crashing or track departure. The controller that makes better choices can certainly win the race.

2.3 Artificial Potential Fields (APF)

Artificial Potential Fields (APF) is an algorithm developed by Khatib in 1986 [8]. Its main application in robotics is path finding and obstacle avoidance. Since its development and until now, APF gained a lot of attention from researchers and it has been widely studied due to its simplicity and elegance [4].

The main idea is that a surface on which the bot is moving is divided into two types of influence points, with attractive and repelling forces. An obstacle imposes a repelling force while a goal attracts the controller with an attractive force. The controller tries to avoid the repelling forces and moves towards the attractive forces which results in a collision free path towards the goal. This method has also been used successfully in RTS games [7].

Artificial Potential Fields (APF) is used for avoiding obstacles by finding a collision free path. But since there are different types of obstacles and goals, different potentials fields can be initiated for a single controller. A traveling car has two types of obstacles, track edges, and other cars. There might be other types of obstacles/goals that are interesting for the controller, such as the minimum curvature to maintain the highest speed. Points on the track that fulfills the minimum curvature impose attractive force on the controller, while other points which does not fulfill it impose repelling forces.

Although the implementation of APF in autonomous vehicle controllers became very popular, it still requires constant improvement. The performance requirements change and become harder to meet, and so the ability of optimal performance must be insured through the use of optimization methods which can tune up the different parameters, giving the controller more efficiency and robustness [10].

2.4 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is an optimization technique that was originally introduced by Kennedy and Eberhart in 1995. PSO is a technique that delivers computational intelligence using aspects of social interaction instead of purely cognitive abilities. It is inspired from the way birds search for food. Birds travel in swarms and communicate information about food locations in the area, this communication of food locations enables birds to travel towards the best locations where they are more likely to find food [5].

The main concept of PSO is to have a collection of candidate solutions which coexist and cooperate to solve an optimization problem. Each candidate solution is called a “particle”, and there are a number of particles searching the solution space for an optimal solution similar to the way a swarm of birds search for food. There are two influences on the movement of the particle, the global best value and the local best value. A particle adjusts its position in the solution space according to these two driving influences.

Initially a PSO model consists of a number of particles randomly distributed in the search space of a problem. Each particle evaluates the fitness function at its current location and communicates it with the rest of the swarm. To move towards improving locations, the particle calculates its movement by comparing its local best value with the global optimal value in the entire swarm to determine the direction of improvement. Further iterations takes place after all particles have been

moved and eventually the swarm as a whole would close to an optimal solution [11] .

While the problem when using optimization techniques in an online setting is long calculation time, this is not the case with PSO. Using PSO is suitable under strict performance requirements because of its low computational cost and computational complexity [12].

PSO models are highly dynamic in nature, which gives them a great advantage. It is possible to tune the model to suite the problem at hand by controlling a number of variables which constitute the model, such as the number of iterations, step size, and number of particles. The effect of adjusting these variables is increasing and decreasing the computational overhead. Another advantage of using PSO is the possibility of training the particles to enhance their performance, trained particles have less communication overhead and faster convergence to an optimal solution [13].

One of the benefits of using PSO is that it offers a great potential to overcome the shortcomings of APF when dealing with areas of local minima and trap situations, since the particles are initially distributed randomly before they start converging towards optimality, which covers different areas of the solution space and insures that they do not get stuck at a local minima. Also, particles travel in the solution space while having constant communication with other particles which gives them the ability to constantly keep moving until the global optimum is reached. This highly dynamic behavior makes particles break easily from local minimal solutions into better improving solutions within the solution space. Because particles move in different directions towards optimality, they have the ability to abandon a local best solution to solutions of lower fitness value on their way to converge to solutions with better fitness values.

2.5 TORCS Environment

The Open Racing Car Simulator (TORCS) is a visual car racing simulator used by researchers to develop new autonomous car controllers. The first reason for choosing TORCS is that it allows us to select which algorithm to use in developing the controller and to set different opponents and tracks combinations to test our controller with. The second reason for choosing TORCS is that it reconstructs the car racing environment in a very realistic way, TORCS takes into consideration the physics of driving a car such as traction, aerodynamics, track status, car damage, it allows having opponents, collisions, and all the different aspects of a real race [6]. Games similar to TORCS does not serve us very well because either it does not have the same realistic environment level like TORCS, or it focuses on improved 3D scenery and fun aspects like extra drifting which creates a high computational load that is not necessary, and it reduces the level of accuracy of the results. Some of these games are Trigger Rally, SuperTuxKart, Speed Dreams, and VDrift.

Controllers in the TORCS environment can be developed using any possible method, allowing researchers to test the efficiency of various algorithms and to later improve their implementation. There is a collection of tracks ranging from narrow and curvy to wide long stretching speedways, this track variation is important to test the controller's performance in different situations.

To move the car, the controller has to communicate with the TORCS server, the communication process includes receiving sensor data and sending control data to the server. The TORCS server provides a number of sensors that describes the current status of the controller's environment, the controller then calculates new control values and send them back to the server to execute them in the next step.

The following table shows the available sensors in the TORCS environment [14] .

Name	Range (unit)	Description
angle	$[-\pi, +\pi]$ (rad)	Angle between the car direction and the direction of the track axis.
curLapTime	$[0, +\infty)$ (s)	Time elapsed during current lap.
damage	$[0, +\infty)$ (point)	Current damage of the car (the higher is the value the higher is the damage).
distFromStart	$[0, +\infty)$ (m)	Distance of the car from the start line along the track line.
distRaced	$[0, +\infty)$ (m)	Distance covered by the car from the beginning of the race
focus	$[0, 200]$ (m)	Vector of 5 range finder sensors: each sensor returns the distance between the track edge and the car within a range of 200 meters. When noisy option is enabled, sensors are affected by i.i.d. normal noises with a standard deviation equal to the 1% of sensors range. The sensors sample, with a resolution of one degree, a five degree space along a specific direction provided by the client (the direction is defined with the focus command and must be in the range $[-\pi/2, +\pi/2]$ w.r.t. the car axis). Focus sensors are not always available: they can be used only once per second of simulated time. When the car is outside of the track (i.e., pos is less than -1 or greater than 1), the focus direction is outside the allowed range ($[-\pi/2, +\pi/2]$) or the sensors has been already used once in the last second, the returned values are not reliable (typically -1 is returned).
fuel	$[0, +\infty)$ (l)	Current fuel level.
gear	$\{-1, 0, 1, \dots, 7\}$	Current gear: -1 is reverse, 0 is neutral and the gear from 1 to 7.
lastLapTime	$[0, +\infty)$ (s)	Time to complete the last lap
Opponents	$[0, 200]$ (m)	Vector of 36 opponent sensors: each sensor covers a span of $\pi/18$ (10 degrees) within a range of 200 meters and returns the distance of the closest opponent in the covered area. When noisy option is enabled (see Section 7), sensors are affected by i.i.d. normal noises with a standard deviation equal to the 2% of sensors range. The 36 sensors cover all the space around the car, spanning clockwise from $+\pi$ up to $-\pi$ with respect to the car axis.
racePos	$\{1, 2, \dots, N\}$	Position in the race with respect to other cars.
rpm	$[2000, 7000]$ (rpm)	Number of rotation per minute of the car engine.
speedX	$(-\infty, +\infty)$ (km/h)	Speed of the car along the longitudinal axis of the car.
speedY	$(-\infty, +\infty)$ (km/h)	Speed of the car along the transverse axis of the car.
speedZ	$(-\infty, +\infty)$ (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 noisy option is enabled (see Section 7), sensors are affected by i.i.d. normal noises with a standard deviation equal to the 10% of sensors range. By default, the sensors sample the space in front of the car every 10 degrees, spanning clockwise from $+\pi/2$ up to $-\pi/2$ with respect to the car axis. However, the configuration of the range finder sensors (i.e., the angle w.r.t. to the car axis) before the beginning of each race. When the car is outside of the track (i.e., pos is less than -1 or greater than 1), the returned values are not reliable.
trackPos	$(-\infty, +\infty)$	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.
wheelSpinVel	$[0, +\infty]$ (rad/s)	Vector of 4 sensors representing the rotation speed of wheels.
z	$[-\infty, +\infty]$ (m)	Distance of the car mass center from the surface of the track along the Z axis.

Table 1: TORCS sensors

The following table shows the control variables which are used to control the movement of the car [14] .

Name	Range	Description
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,...,7	Gear value.
steering	$[-1,1]$	Steering value: -1 and +1 means respectively full right and left, that corresponds to an angle of 0.785398 rad.
focus	$[-90,90]$	Focus direction in degrees (see the focus sensors in the previous table)
meta	0,1	This is meta-control command: 0 do nothing, 1 ask competition server to restart the race.

Table 2: TORCS control variables

Each car can get damaged when it collides with other cars or when it goes out of track and hit obstacles on the side of the road, the server reduces the car speed when it acquires a high level of damage, and it can eventually terminate it before the end of the race.

2.6 Related Work

2.6.1 Robot and Vehicle Controllers

In the area of robotics, O. Khatib proposed the concept of Artificial Potential Fields (APF) as a real time path planning and obstacle avoidance approach [8]. This approach is dependent on the position of the mobile robot relative to the goal position and obstacles in the operational space. According to the potential fields principle, the controller moves in a field of forces where the position to be reached is the attractive force, and the obstacles are repulsive forces to the controller. The sum of the attractive and repulsive forces determines the direction of movement for the robot. Although the use of APF provided a collision free path in a simple straight forward manner, the problem of local minima can occur in some situations. Local minima means that the bot gets stuck in the same position due to even repelling and attractive forces.

Koren and Borenstein [4] developed a method called Virtual Force Field (VFF) which is based on APF. They have experimented with it using their robot CARMEL (*Computer-Aided Robotics for Maintenance, Emergency, and Life support*). After running a series of experiments, they have identified the main problems that are inherent in all methods using APF regardless of their specific implementation as:

1. Trap situations due to local minima (cyclic behavior).
2. No passage between closely spaced obstacles.
3. Oscillations in the presence of obstacles.
4. Oscillations in narrow passages.

Due to the drawbacks in APF, Koren & Bornstein have developed a new method for fast obstacle avoidance called *vector field histogram* (VFH), which gave better results than VFF under the same circumstances. However, VFH does not insure global optimality.

S. S. Ge and Y. J. Cui, added another limitation which is goals non reachable with obstacles nearby (GNRON). To solve this problem, they have introduced a new repulsive potential function, by taking the relative distance between the robot and its goal into consideration [15]. The idea of a repulsive function serves very well in organizing the movement and keeping it towards promising solution areas.

Other researchers studied APF and introduced significant improvements to the classical method such as finding solutions to the local minima problem [16]. A method called Evolutionary Artificial Potential Field (EAPF) was introduced as a mobile robot path planning approach that deals with moving obstacles. EAPF uses APF in combination with Genetic Algorithm (GA) [10].

In the area of vehicle controllers Wolf et al [9], aimed at controlling a vehicle in a multilane highway. They aimed at providing high safety by avoiding collisions and preventing unwanted lane departures. The method used was APF, they have created a potential field by combining a number of potential functions for track keeping, lane keeping, speed preference, and obstacle avoidance. However, their method is meant to keep the vehicle in the center of the lane, which might not always be an optimal choice. According to Wolf, the problem of local minima which exists in robot navigation has a positive effect on the traveling vehicle, since it keeps it behind leading vehicles.

In the area of flying vehicles, a work done by GU et al, addresses the problem of trajectory optimization for a flying object in a three dimensional space. An algorithm was introduced by

combining PSO with APF to reduce self exposure and enemy detection threats. The target function was designed to dodge obstacles and improve the survival probability. By applying APF principles, particles were designed to adjust their movement according to the resulting angle of different forces applied in the environment, which resulted in a more effective particle movement. In the experiment a comparison between PSO and APF-PSO was held, and the results showed that using APF-PSO improved the detection accuracy and reduced the particle convergence speed. Based on their conclusions, the use of APF-PSO reduced the time needed to reach optimality, and helped in avoiding local minimum situations [17].

Another research done by S. Hassan and J. Yoon, addresses the problem of navigating a swarm of nano drug carriers in a drug delivery system using PSO. The proposed algorithm (NPSO) proved to generate more optimal or near optimal solutions than the traditional PSO with lower computational cost and better convergence speed [18].

2.6.2 Controllers in Games

APF was applied in areas other than robotic motion planning, Johansson and Hagelbäck applied APF in Real Time Strategy (RTS) games where it proved to be a good solution for group motion in a dynamic changing environment [7].

Agapitos et al, modeled the game environment using stateful genetic algorithms to help the controller understand the game environment and plan actions more effectively. They have shown that using a model building behavior along with controlling behavior can help controllers achieve better results in the racing track [19]. Although their conclusion might be true, it will definitely make it harder to meet the performance requirements because modeling the environment requires extra processing before the controlling actions are calculated.

Another approach is using Neural Networks (NN) for building autonomous controllers and teaching them the driving behavior of other bots. Athanasiadis et al, proposed a controller that learns progressively starting from a simple rule based controller and improving its decisions by training the NN. They concluded that this method results in increased controller performance [1]. However, there are some limitations when using NN in such a highly dynamic environment, since the computational overhead of the NN is high and might result in delayed decision making. Another issue is the limited training data which might be insufficient or lacking important racing skills.

Cardamone et al, designed a controller using genetic algorithms. In their work, the racing track was divided into segments and the best car racing line was searched using a genetic algorithm. The selected racing line represents a trade off between the minimum curvature and the shortest racing line on each segment [2]. However, dividing the racing track into segments does not reflect a reactive behavior since it requires prior knowledge of the track shape which might not be possible all the time.

Designing autonomous controllers based on APF have been done by Uusitalo and Johansson, who have built a controller named “Powaah” [20]. “Powaah” participated in the CIG 2011 and scored 6th out of 10 competitors [3]. The authors used APF as a path planning method and created a Multi Agent System (MAS) as a structure to facilitate the use of APF in achieving various tasks during the race such as avoiding collisions and overtaking opponents.

The MAS of the “Powaah” controller consists of the following agents:

- The Driver Agent
- The Shortest Path Agent
- The Curvature Agent
- The Track Agent
- The Navigator Agent
- The Interface Agent

Each one of those agents contributes in a different way to achieve the controller's main goal of winning the race. The concept of assigning various charges to perspective positions on the track was used in “Powaah” along with the concept of Look Ahead (LA) points. We have used the same concepts when designing our controller, and we will talk more about these concepts and all the used formulas in the controller section. The reason for using “Powaah” as the base of this work is the use of APF and MAS in building the controller. Using a MAS serves the modularity of the driving tasks, and using APF ensures a collision free path with low computational expense.

3 RESEARCH METHODOLOGY

3.1 Research Design

To answer the aforementioned research questions, we decided to do an experiment using the TORCS server as a simulation tool. Simulation is a suitable approach in our case since it helps us measure the changes in different performance variables and understand the relationship between variables. By conducting a simulation, it will eventually be possible to conclude if the proposed algorithm is actually successful in currying out the tasks for which they have been designed at the desired level of performance.

The first component we need to identify is the participants in this experiment. The first and most important participant is the APF-PSO controller, which is used also as a APF controller when disabling the PSO. The other controllers or opponents are selected carefully from the top performing controllers in the CIG competitions and from the TORCS server. Five opponents were selected, those controllers are built based on different approaches and they showed good results when we tested them on TORCS.

The second component is the variables in the experiment. There are two types of variables in an experiment; independent, and dependent. Independent variables are the ones we decide to set their values, and they affect the values of dependent variables. Dependent variables are also called performance variables and they are the ones we use to measure the level of performance for any controller. Our independent variables are:

- Number of opponents
- Length of the race (i.e. number of rounds)
- Track shape (i.e. width, number of turns)

Our dependent variables are:

- Finishing rank
- Total time
- Best lap time
- Top speed
- Total damage

We began by implementing the proposed algorithm as a controller using a skeleton client code provided by Loiacono et al [14]. The skeleton provided means of communication with the TORCS server and an initial dummy driver that could only drive along the middle line of the track. When our APF-PSO was fully implemented in the client, it was used in two experiments to race solely and against other controllers in the TORCS server.

Since the performance is dependent on whether we are driving alone or racing against other opponents, it was necessary to conduct two experiments with different configuration each time. The first experiment was solo driving on four different tracks 10 laps each, where we compared the APF controller against the APF-PSO controller to see if using PSO does actually increase the speed and reduce the racing time and damage. The second experiment was a racing competition between our APF-PSO controller and five other controllers, one of them is an APF based controller (i.e. Powaah), the aim was to measure the performance variables that was used in the analysis and assessment of the controller to finally come to a conclusion if our APF-PSO controller does achieve competitive results in a racing simulation.

We have used the Formula 1 (F1) ranking system which is used in the Simulated Car Racing Championships (CIG) [3]. According to this system each controller is given a total number of points based on its ranking:

1 st rank:	10 points
2 nd rank:	8 points
3 rd rank:	6 points
4 th rank:	5 points
5 th rank:	4 points
6 th rank:	3 points

We used this ranking on all performance variables and concluded which controller was best in the results of each variable. Finally, all the results are discussed to draw the final conclusions and answer the research questions.

4 MODELING THE CONTROLLER

4.1 Proposed Method

Based on the identified requirements for building an autonomous racing car controller, which are the ability of achieving car movement, maneuverability, and maintaining top speed. And based on the fact that the TORCS server must receive control instructions from our controller in a timely manner, adding the need to find the optimal racing line, we concluded that we need to have an algorithm that is both efficient in car racing, ensures optimality, and does not create massive calculation overhead.

APF is a light weight algorithm, it creates a collision free path simply by assigning positive and negative charges and does not require heavy computations. Similarly, PSO is a light weight algorithm that offers computational intelligence in a simple approach far from complexity. Both algorithms are suitable to be used in a racing controller, APF for path finding and PSO for racing line optimality.

Since both APF and PSO are simple straight forward approaches that satisfies the aforementioned requirements, the proposed method is to build our controller based on using APF for path finding along with PSO to evaluate possible moves and choose the optimal ones. Our proposed method is called *Artificial Potential Fields with Particle Swarm Optimization* (APF-PSO). By combining APF and PSO our controller benefits from the simplicity of APF and the high flexibility of PSO.

Machine Learning (ML) approaches such as Neural Networks (NN) and Decision Trees (DT) create high computational overhead which makes them hard to use in an online setting, they also require a lot of expert training to reach acceptable results [19]. Optimization Techniques such as Genetic Algorithms (GA) are also computationally expensive, there is a large difference between the shortest and the longest solution time which makes it unreliable in online and real time applications. Also, there is no assurance that the algorithm will find a global optimum solution [21].

4.2 Controller Design

Our controller uses the same organizational principles used by the “Powaah” controller described by Uusitalo and Johansson [20]. We are going to use the same Multi Agent System (MAS) structure, and the same Look Ahead (LA) points approach for controlling our car. In this section we will describe the organizational scheme and formulas used in the controller.

To start the mathematical modeling of our controller we need to identify the inputs, outputs, and the operations in between.

The input will be sensor data from the TORCS server. There are specific sensors which we are interested in, these sensors are:

- Angle: returns the angle between track and car axis.
- Track: returns the distance between car and track edges.
- Opponent: returns the distance between car and opponents.
- SpeedX: longitudinal speed.
- Gear: current gear.
- RPM: current engine Round Per Minute (RPM).

The output variables are control variables that will be calculated and passed to the TORCS server which will execute them, and as a result the car status will be updated. These output variables are:

- accel: applies car acceleration.
- brake: applies car brakes.
- gear: changes the current gear.
- clutch: applies clutch.
- steering: updates the steering angle.

The controller receives the input values and applies APF and PSO in calculating new values for the output control variables. To achieve its task, the controller has to create the different fields which works as a conceptual representation of the forces in the environment around the car, then the controller finds the possible moves and finally choose the optimal one.

Lets go into more details about how the controller works after receiving the input values from the TORCS server until passing the control variables back to the server:

1- Creating fields and assigning charges

There are a number of fields that a racing car controller would be interested in. Each field represents a distinct objective such as avoiding collisions or overtaking opponents. Fields are populated with charges that are specific to it. Charges represent the attractive and repelling forces. They are used by the controller to calculate how much potential there is for every field. Charges are either attractive or repelling, so they are assigned positive values or negative values depending on their type. Each charge has a magnitude and a direction, the magnitude is calculated by the controller depending on the distance and the type of obstacle or goal we are looking at. More about the assignment of charges will be discussed when we talk about different agents.

2- Particles and lookahead points

After assigning the charges the next step is finding the best next move, to do that we distribute particles on random positions ahead of the car, these positions can be called Look Ahead (LA) points. Based on the work of Uusitalo and Johansson [20] , LA points represents a combination of possible control actions. Those actions include the steering angle, acceleration, and breaking values which the controller commits within the current time frame, and by executing these control actions, the car is expected to reach the position of the LA point in the next frame. LA points are evaluated using a fitness function which calculates the potential at each point to find the point with the highest potential and use it as the optimal solution in the current frame.

As our particles travel in the solution space, we create corresponding LA points and evaluate them. This approach reduces the overhead of creating and visiting a pre-specified space of LA points which might be huge. On the other hand it does not restrict us to a limited number of LA points since we can create as many points as needed.

During the search of an optimal solution, each particle evaluates its position by adding potentials from different fields, then a global best value will be declared in the entire swarm of particles, by comparing their fitness values to the global best, particles decide to move towards the optimal LA point or to stay in their positions if their fitness value is equal to the global best value. The range of movement is $[-45,45]$ degrees because the steering angle is only limited to 45 degrees to the left or to the right of the car access. Also there is an upper limit and lower limit for how far LA points can be placed, these limits can be calculated using the following table and set of formulas.

The following table represents the corresponding acceleration when applying full throttle or full brake.

Gear	Acceleration (m/s ²)
1	7.1666667
2	6.045
3	4.8783
4	4.05
5	2.81
6	2.04
-1	-21.16

Table 3: Resulting acceleration when applying full acceleration or full brakes

Using the values from Table 3, and based on the current gear, we can calculate the acceleration at the upper and lower limits using the formula:

$$acc_{Limit} = acc_{gear} \cdot limit_{throttle} + acc_{deceleration} \cdot limit_{brake} \quad (1)$$

To get the acceleration at the upper limit we can substitute with the acceleration corresponding to the current gear from Table 3, and to get the acceleration corresponding to the lower limit we substitute with the -1 gear value which represents acceleration when applying full brakes.

The speed at the upper or lower limit can be calculated using the formula:

$$v_{Limit} = v_{current} + acc_{Limit} \cdot dt \quad (2)$$

The formula for calculating the distance of the limit is:

$$d = v_{Limit} \cdot dt \quad (3)$$

Where d is the distance between the car position and the upper or lower limit, v is the speed at LA, and dt is the time difference between the current frame and the next frame.

After we have calculated the upper and lower limits, and set the range of steering to $[-45, 45]$, the particles are initiated and moved within the given solution space.

Particle movement is a process that consists of three steps:

- Determining the direction of movement
- Creating the corresponding LA point in the new position
- Moving to the new LA point

The direction of movement is determined with reference to the position of the particle at the global best value. Each particle saves a vector $\vec{V} = (d, a)$ as its current position where d is the distance

between the car and the position of the LA point, and a is the angle between the car access and the LA point.

When a particle is located at the global best value, its location is passed to the other particles which then calculates the difference between its current position and the global best position. Suppose we have the global best at the position $\vec{V}_{Global\ Best}=(20,10^\circ)$, a particle n at position $\vec{V}_n=(15,5^\circ)$ has to update its position one step towards the global best position. We have determined the step length to be $\vec{V}_{Step\ Length}=(1,1^\circ)$ meaning that every time a particle updates its position it moves 1° and 1 meter. In our example the particle n will update its position to be $\vec{V}_n=(16,6^\circ)$.

To create the corresponding LA point at the new movement position there are a number of variables that needs to be calculated, as we are going to do in the following set of formulas.

$$v_{LA}=d \div dt \quad (4)$$

$$acc_{LA}=(v_{LA}-v_{current}) \div dt \quad (5)$$

$$p=acc_{LA} \div acc_{gear} \quad (6)$$

Where p is the value of throttle that we need to apply to get to the LA point. And since we have the angle of the LA point we can build the corresponding control string with values of steering, throttle and braking.

Particles start converging towards the optimal position step by step, each time calculating the best value position and moving one step towards it. On their way to optimality, particles evaluate new positions and might find a better solution which then becomes the new optimal position. When the global optimum is found it is sent to be executed by the TORCS server.

Due to the dynamic nature of car racing, we have ignored the effect of historical self best found solution on the particles, this implies that we are only using the current self best against the optimal solution found in the entire swarm.

The number of particles we are using are 30 particles, and we iterated them a maximum of 100 times to get an optimal solution. This configuration can be tuned to increase the performance, however it might increase the overhead and make our controller stop responding to the TORCS server.

4.3 The Multi Agent System

To build our controller, we are going to use a Multi Agent System (MAS) as a main structure. There are two benefits of using the MAS approach in designing the controller. The first one is the suitability of a MAS to the nature of the racing environment, since our goals and tasks are modular, it fits well to use different agents to achieve them. The second reason for using MAS is suitability to the APF approach, since we have a number of fields, it is good to have different agents to deal with each one of them.

To successfully drive a car on a racing track against other opponents there are many tasks that the controller has to carry out simultaneously and in a timely manner in order to achieve good results in the race. If we consider that behind each of these tasks is a goal, and we have a number of goals to be achieved simultaneously, the MAS provides a mechanism to work with different goals better

than sequentially addressing a list of goals. A MAS approach is modular by nature, which means that it can handle a scattered problem better than other approaches.

As we have discussed earlier, the APF approach provides a way to deal with different types of obstacles, and helps find a path between them. It can also be used to reach different goals by defining a field of attractive force for each one of these goals. A MAS can handle this by defining an agent for each field, and letting that agent handle the negative and positive forces that affect the car. Finally we can establish cooperation between the agents in order to find the desired path taking into consideration the feedback from all agents.

In our controller we have defined a MAS consisting of a cooperating agents, and they make up the backbone of the controller. Below is a list of all the agents and a description of their role in the MAS.

Track Agent

The first and most important agent is the track agent. As the name suggests, the track agent tries to keep the car on track. The way it achieves that is by assigning positive charges to the middle of the track. This results in the forward movement of the car, at the same time maintaining the car on the track.

The way this agent calculates the middle of the track is by using this formula:

$$m = a - (p/2) \quad (7)$$

Where m is the steering angle to the middle of the track, a is the angle between car axis and track axis, p is the car current track position.

Overtaking Agent

The work of the overtaking agent is a bit tricky, since overtaking of opponents is a process that requires suitable conditions for overtaking otherwise it might result in either a collision or track departure. The agent assigns positive charges to the right or left of an opponent. The formula used to calculate the charges:

$$pc(j) = j \cdot 10 \quad (9)$$

$$j = i \pm [1, 2, 3]$$

Where pc is the positive charge function, i is the opponent position, j is overtaking directions \neq opponent position.

Speed Agent

The speed agent forces the car to follow the longest line of sight that is free of obstacles and keep the highest speed possible. This agent assigns a positive charge to the direction of the longest line of sight, the longer the line of sight the stronger the positive charge, and vice versa.

The formula used to calculate the charges:

$$pc(i) = d \cdot 10 \quad (10)$$

Where pc is the positive charge function, i is the sensor direction, d is length of line of sight.

Opponent Agent

The rule of the Opponent Agent is very important, it prevents collisions with opponent cars. Opponent agent assigns negative charges to the direction of opponent cars. The magnitude of the

negative charge depends on how far is the opponent.
The formula used to calculate the charges is:

$$nc(i) = (-10) \cdot d \quad (11)$$

Where nc is negative charge function, i is direction of the opponent, d is the distance to opponent.

Opponent agent and overtaking agent can be seen to play a complementary rule, since the overtaking agent assigns positive charges to possible takeover areas to the left and right of the opponent and the opponent agent assigns negative charges to prevent collision with the opponent.

Curvature Agent

One of the most important rules during the race is carried out by the curvature agent. This agent tries to maximize the radius of the curve [20] which helps in maintaining the highest speed while turning in a curve, it also helps in increasing the control and prevents sliding out of track by reducing the sharpness of the turn which reduces the effect of the centrifugal force that pushes the car to the opposite side of the turn.

What the curvature agent does is that it places a positive charge to the opposite side of the upcoming turn, this results in keeping the car away from the turn until it is close enough to start turning. The range at which we start assigning the positive charge is 150 meters up to 50 meters before the turn. The formula that it used to assign the positive charges is:

$$pc = 10 \cdot (1 - d \div w) \quad (12)$$

Where pc is the positive charge, d is the length to the curvature, w is the width of the track.

PSO Agent

The rule of the PSO agent is to create the particles and move them to find the best next move. Particles are initiated randomly, they evaluate the position they are on based on the potential it has. After evaluation, a global best value is declared and particles start moving towards the direction of that global best value, after one step of movement, a possible new global value might be declared, which will cause further movement towards it. Each time, particles compare their own best value with the global best value, and decide whether to move or not. Movement results in visiting more points on the way to optimality which can possibly give an optimal solution.

Driver Agent

The Driver takes the optimal control values from the PSO agent, and communicates it with the TORCS server in order to apply the next move.

The following figure shows the agents of the MAS.

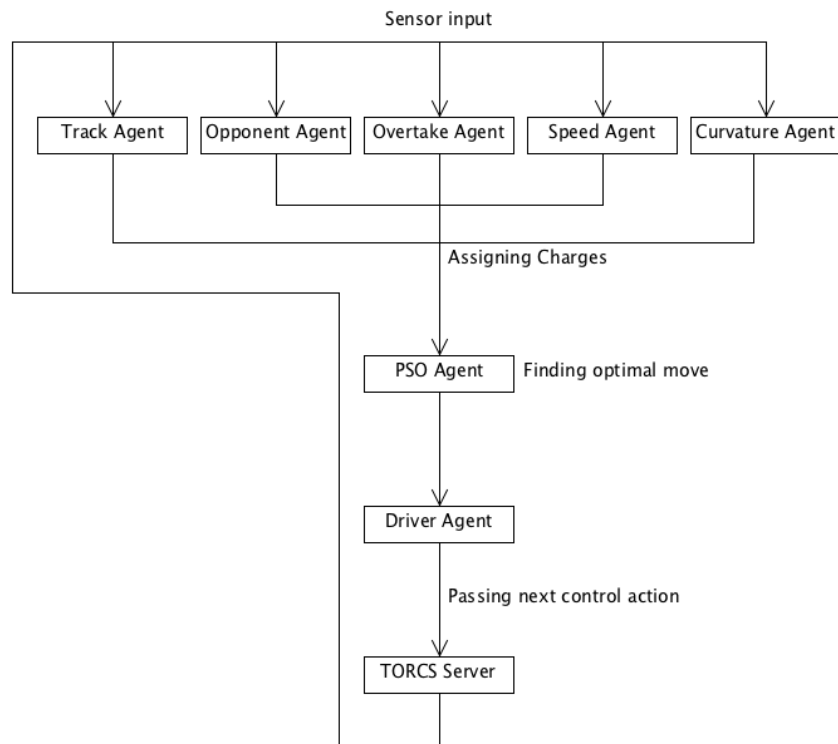


Figure 1: Agents in the MAS

5 EXPERIMENTS

After implementing the APF-PSO controller, we proceed to evaluate it using the TORCS server. We have conducted two experiments, the first experiment is conducted to test the effect of using particles with potential fields, so we tested the controller both with and without particles and compare the results in both cases. The second experiment is done to compare the performance of our controller with other controllers who happened to score good results on the TORCS server including the “Powaah” APF based controller. To achieve good hardware performance during the experiment, the TORCS server is run using a 2.53 GHz Intel Core i5 processor, with 4 GB DDR.

5.1 Variables

The variables used in the experiments are either dependent or independent variables. There is one way dependency relation between these two types of variables. This implies that by assigning some values to the independent variables, we get corresponding outcome in dependent variables. Dependent variables can also be called performance variables. Independent variables in our experiment are:

- Number of opponents
- Length of the race (i.e. number of rounds)
- Track shape (i.e. width, number of turns)

The corresponding dependent variables are used to measure the performance, we chose the output of the following dependent variables to evaluate our controller:

- Finishing rank
- Total time
- Best lap time
- Top speed
- Total damage

5.2 Tracks

Choosing tracks should be done with consideration to a number of issues, first the tracks should represent a combination of different situations during the race (i.e. High speed, curvy, slippery, etc). Second, the width of the track shows how well the controller can keep the car on track, so different track widths are to be selected. And finally the a variation of long stretches and sharp turns to demonstrate each controller's ability to score higher speeds and better lap times. Below is a description of the three tracks which we choose in the experiment:

Track 1

Track name: CG Speedway 1

Description: Quite fast paced track with consecutive turns

Length: 2057.56 m

Width: 15 m



Figure 2: CG Speedway 1

Track 2

Track name: CG Speedway 2

Description: High speed track with sharp turns and narrow sections

Length: 3185.83 m

width: 15 m

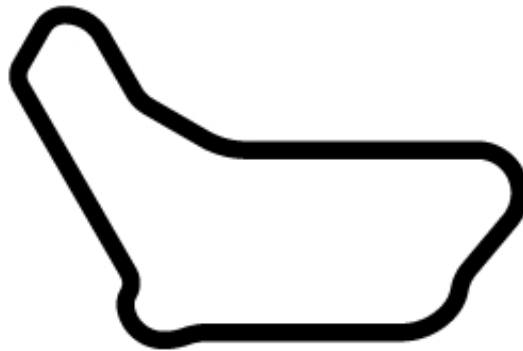


Figure 3: CG Speedway 2

Track 3

Track name: Brondehach

Description: Technically demanding circuit with muddy surface and sharp turns

Length: 3919.31 m

Width: 16 m

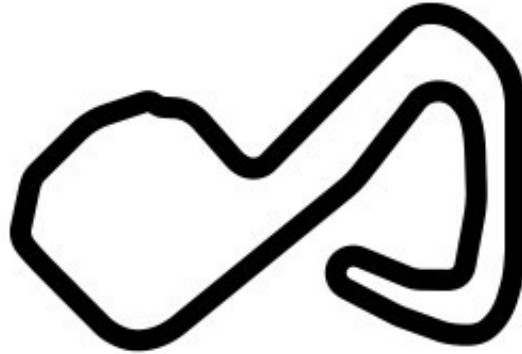


Figure 4: Brondehach

Track 4

Track name: A-Speedway

Description: Rectangle shaped speedway, very wide and tilted inwards.

Length: 1908.32 m

Width: 25 m



Figure 5: A-Speedway

5.3 Opponents

The choice of opponents determines the level of competition we are setting against our controller, for this reason we have chosen five well performing opponents, the main criteria is good performance which we determined by test running all the available controllers both found on the TORCS server and those available online. The selected five opponents are:

Opponent 1: Berniw

Ships with the TORCS server, a very interesting and efficient controller, it uses a vector class for calculating the steering angle, then it uses a heuristic approach to improve the path finding around the track. Collision avoidance is well implemented, which results in low damage compared to other controllers.

Opponent 2: Inferno

Also comes with TORCS, a robust advanced controller which maintains good performance on all tracks and is efficient in overtaking and avoiding collisions.

Opponent 3: BT

Similar to Berniw in structure and also available on TORCS, this controller is a bit more careful, this can be seen in the slightly reduced speed. Overall it has good performance and it is efficient in overtaking and avoiding collisions.

Opponent 4: POWAAH

A good controller that uses a MAS to coordinate the different tasks, with APF as basic path finding approach [20]. However, there are some issues such as high damage and lower top speed compared to other controllers.

Opponent 5: Autopia

A very good controller, it has been a top performing controller for a number of years, and it was the winner of the third leg of the Simulated Car Racing Championship 2010 (CIG-2010). Autopia was built using Genetic Algorithms for optimization of speed and steering [22].

6 RESULTS AND ANALYSIS

6.1 Results

In this section we are presenting the results of the two experiments in tables and figures, further analysis and discussion will come in the following sections.

6.1.1 Experiment 1

As mentioned earlier we will be conducting two experiments, the first one is solo driving to test the effect of using PSO with APF, and the second one is racing against other controllers.

The first experiment is a performance test for using APF-PSO against using APF alone. Solo driving demonstrates the controller's ability in reaching higher performance levels on different tracks without any influences from competing opponents which can increase the damage or decrease the top speed.

The first step is running our controller using APF for 10 laps on each track. The TORCS server gives us the results each time we run on a track, we recorded the results and moved on to the next step which is running the controller using APF-PSO, and again we recorded the results for each track we drove on. The variables which we have recorded are, total time, best lap time, top speed, and damage. Ranking is irrelevant here since we are driving alone. The results of this phase is shown in the following table:

		APF Controller	APF-PSO Controller
Track 1	Total Time	405.09	401.78
	Best Time	44.67	42.46
	Top Speed	247	252
	Damage	0	32
Track 2	Total Time	612.39	559.02
	Best Time	01:03:42	59:47
	Top Speed	268	274
	Damage	0	258
Track 3	Total Time	895.25	886.15
	Best Time	01:36:01	01:28:06
	Top Speed	238	249
	Damage	337	451
Track 4	Total Time	321.21	318.06
	Best Time	28:63	28:20
	Top Speed	256	261
	Damage	0	0

Table 4: Experiment 1 results

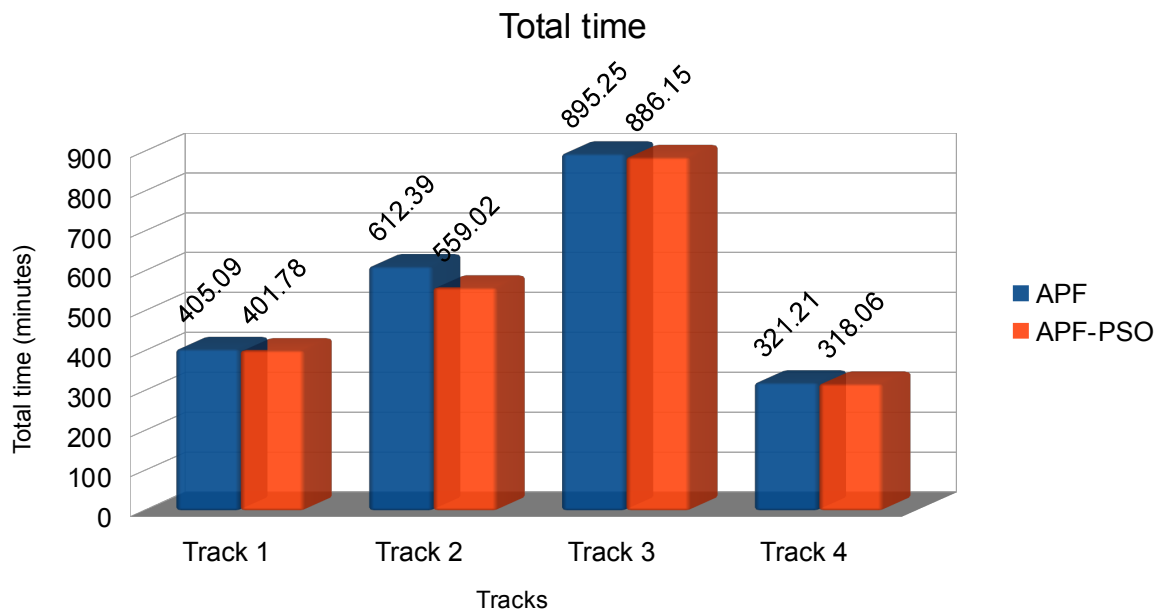


Figure 6: Total time per track

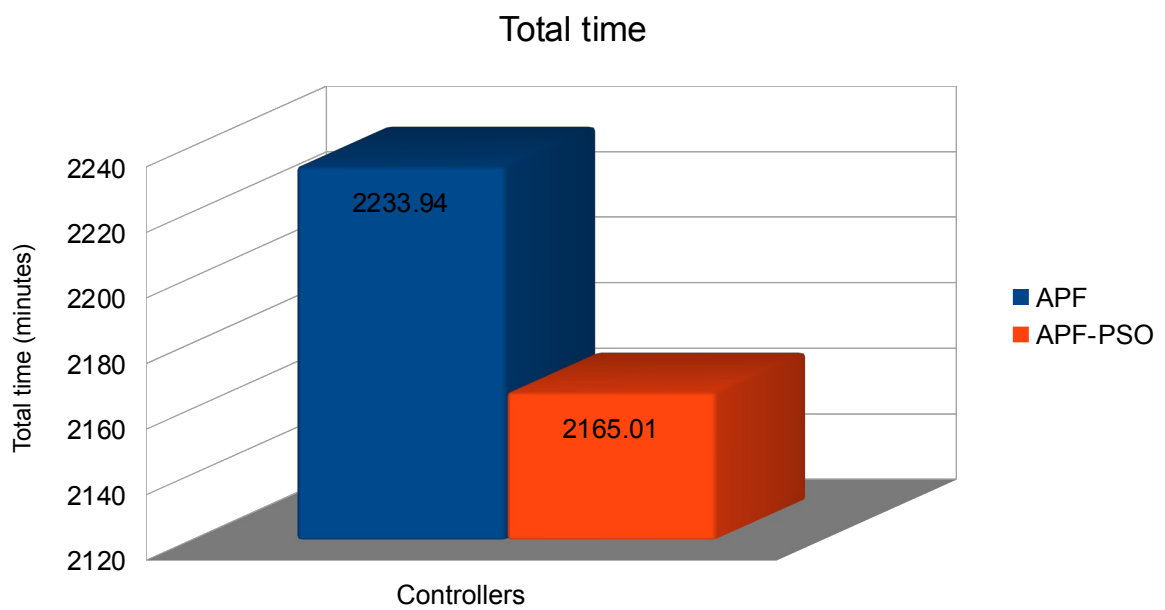


Figure 7: Total time on all tracks

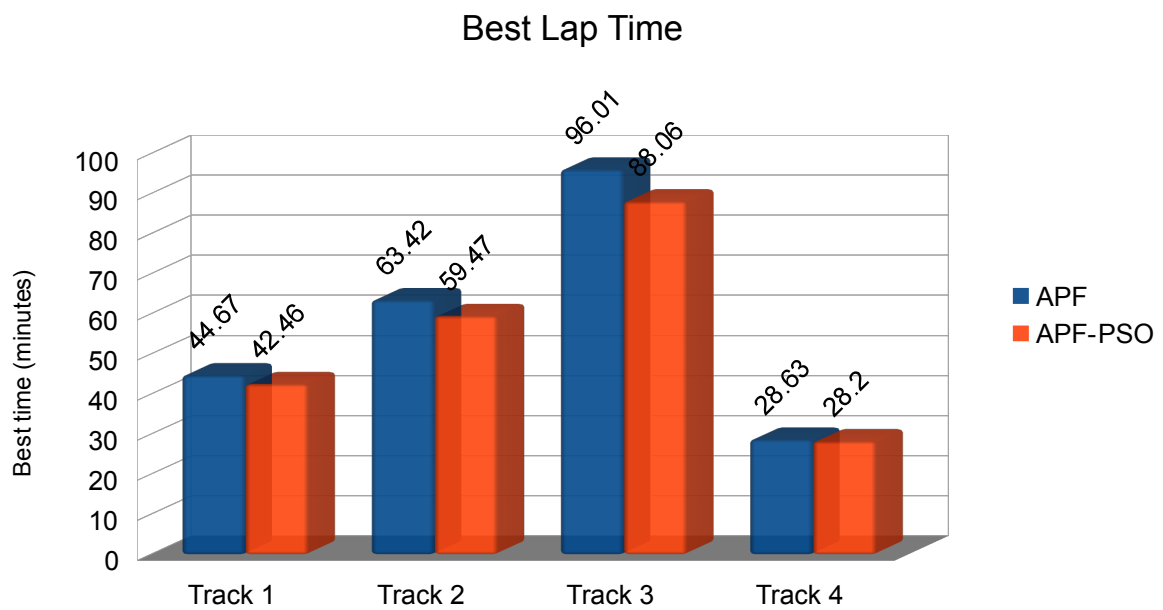


Figure 8: Best lap time per track

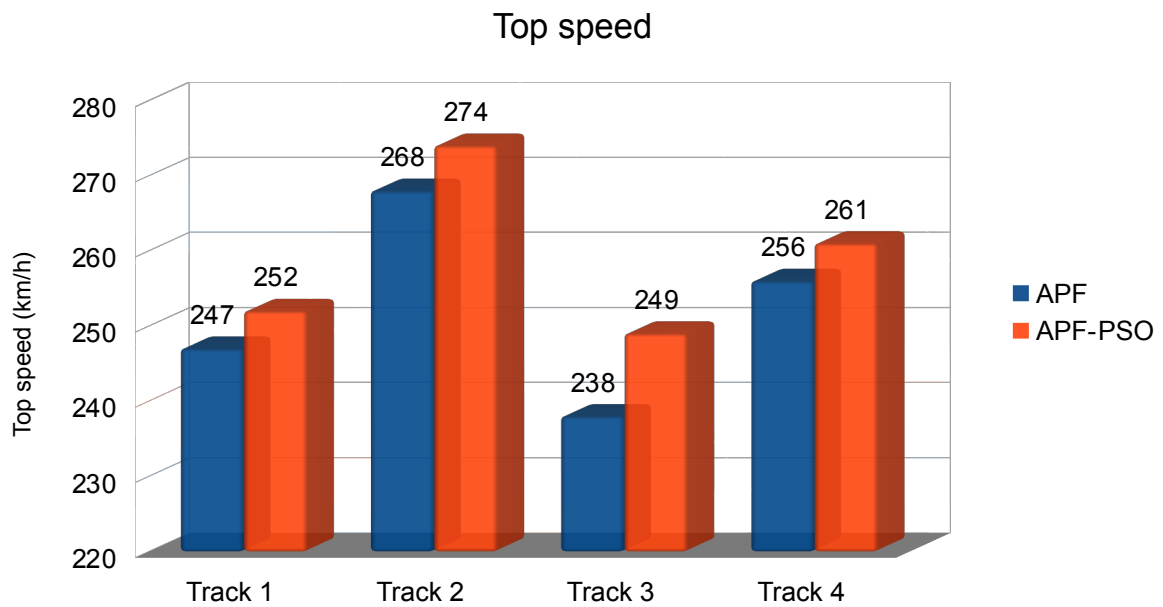


Figure 9: Top speed per track

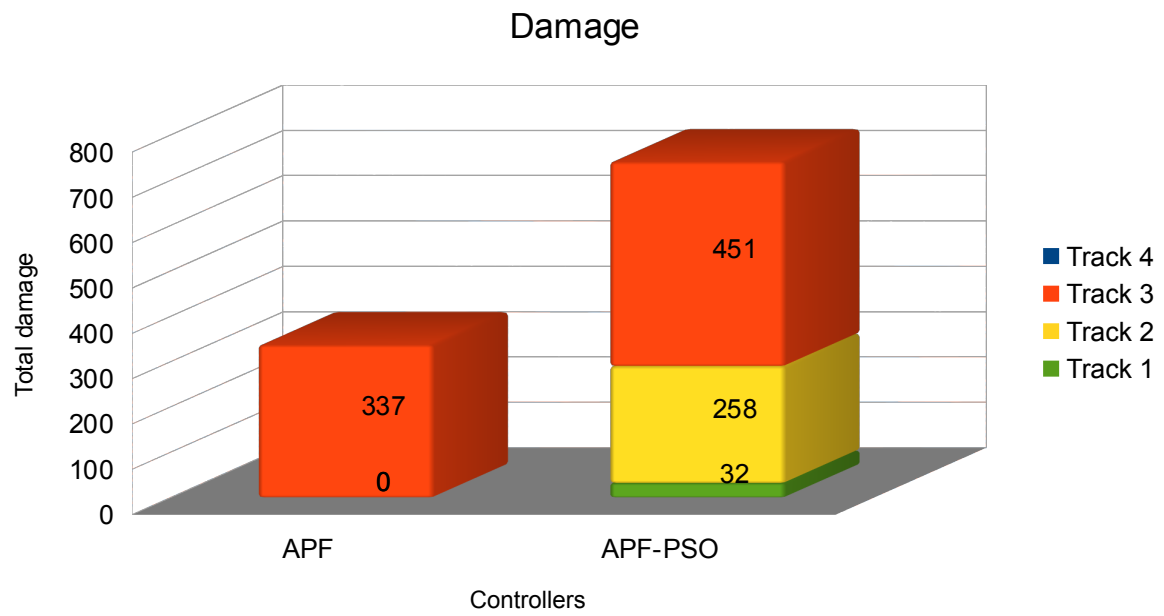


Figure 10: Total damage on all tracks

6.1.2 Experiment 2

In the second experiment we set a series of runs for our controller against the opponent controllers. The results are shown in the following table.

		APF-PSO	Berniw	Inferno	BT	POWAAH	Autopia
Track 1	Rank	1	2	3	4	5	6
	Total Time	403.41	415.32	419.01	422.77	424.66	427.96
	Best Time	40:10	40:96	40:96	41:04	41:37	41:29
	Top Speed	251	247	250	246	241	242
	Damage	236	61	273	1341	0	8
Track 2	Rank	1	4	3	2	6	5
	Total Time	562.06	579.59	576.13	572.83	596.84	580.72
	Best Time	51:54	53:80	53:66	53:23	58:88	55:03
	Top Speed	281	270	278	271	252	262
	Damage	196	27	343	239	82	36
Track 3	Rank	5	1	2	3	4	6
	Total Time	884.09	817.42	831.45	852.37	878.55	887.63
	Best Time	1:29:47	1:20:84	1:21:04	1:23:34	1:27:79	1:29:01
	Top Speed	248	251	251	248	241	243
	Damage	938	107	844	376	1158	349
Track 4	Rank	2	4	3	6	5	1
	Total Time	316.24	320.00	317.17	329.60	323.30	308.13
	Best Time	28:61	28:78	28:76	30:30	31:71	27:41
	Top Speed	260	263	265	259	252	263
	Damage	0	31	29	0	0	16

Table 5: Experiment 2 results

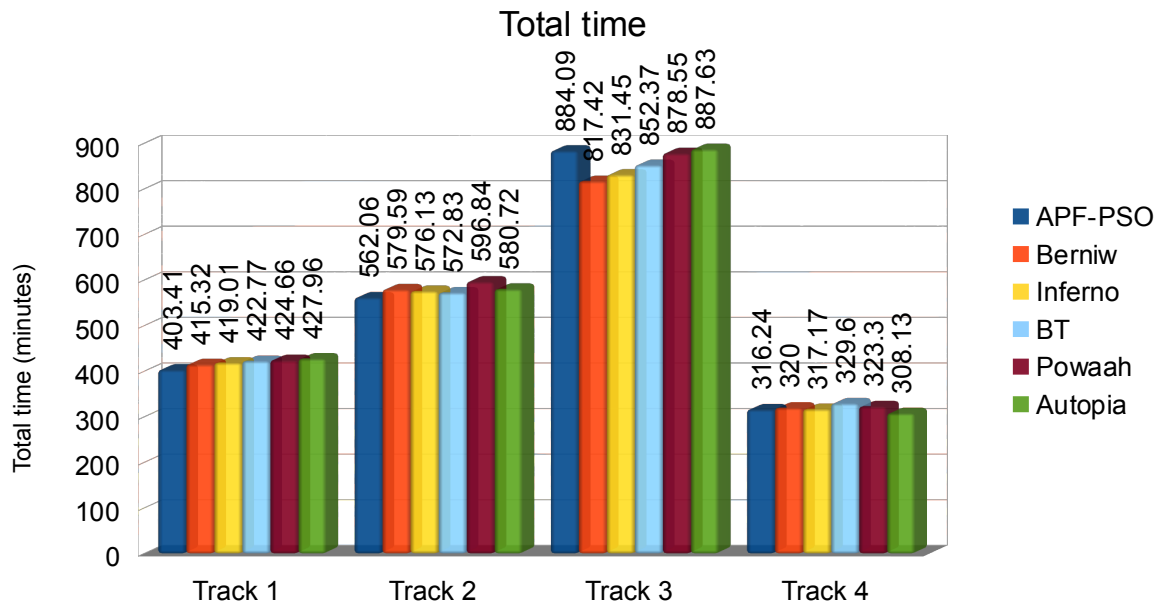


Figure 11: Total time per track

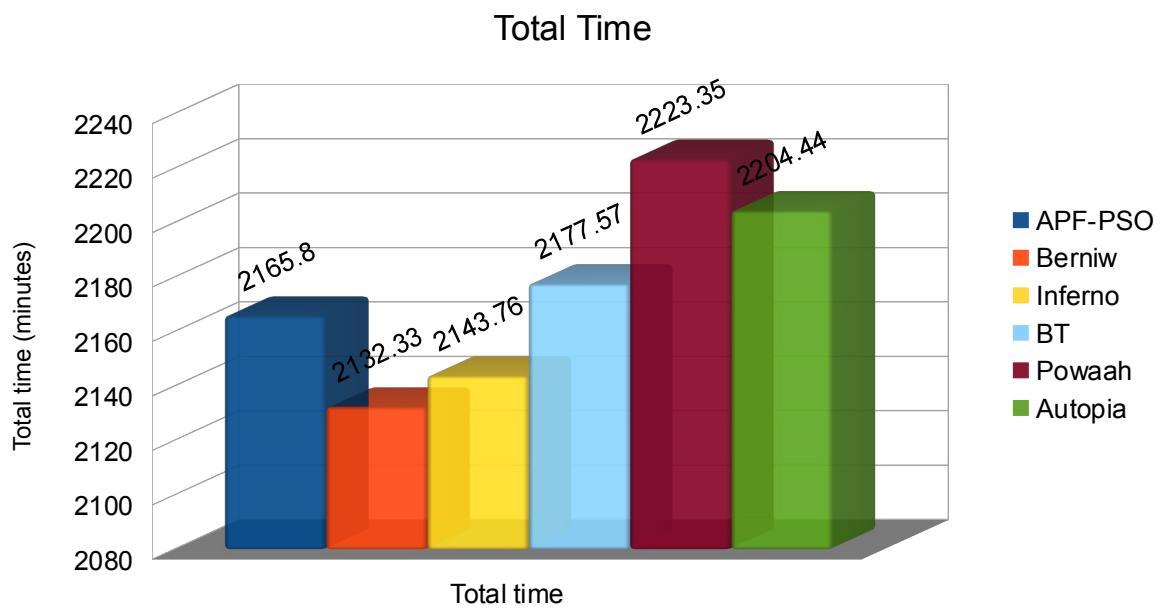


Figure 12: Total time on all tracks

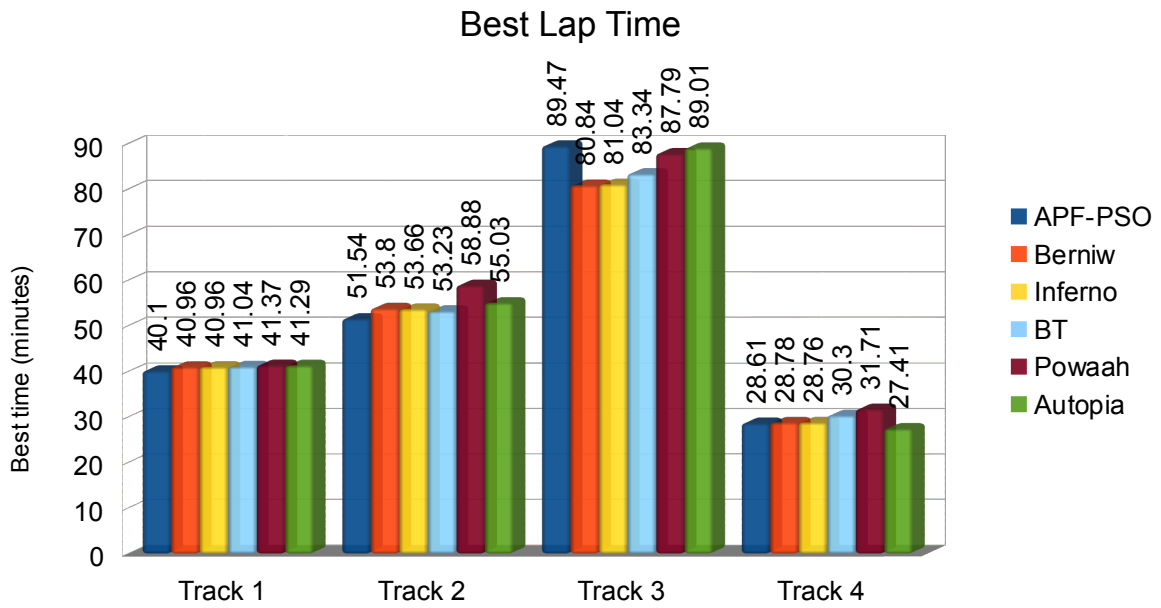


Figure 13: Best lap time per track

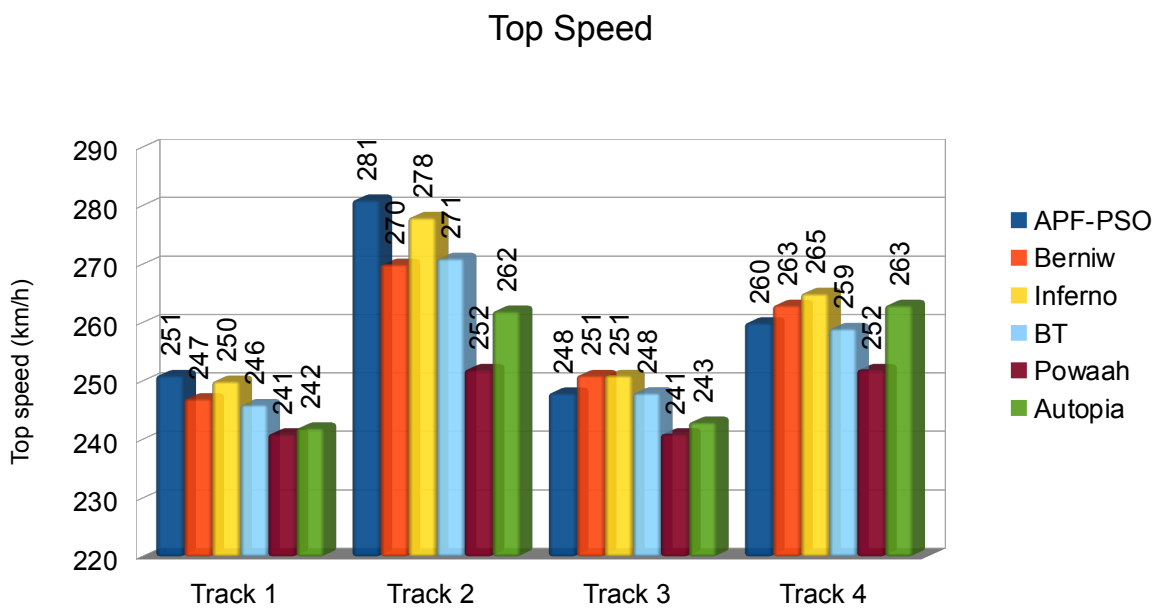


Figure 14: Top speed per track

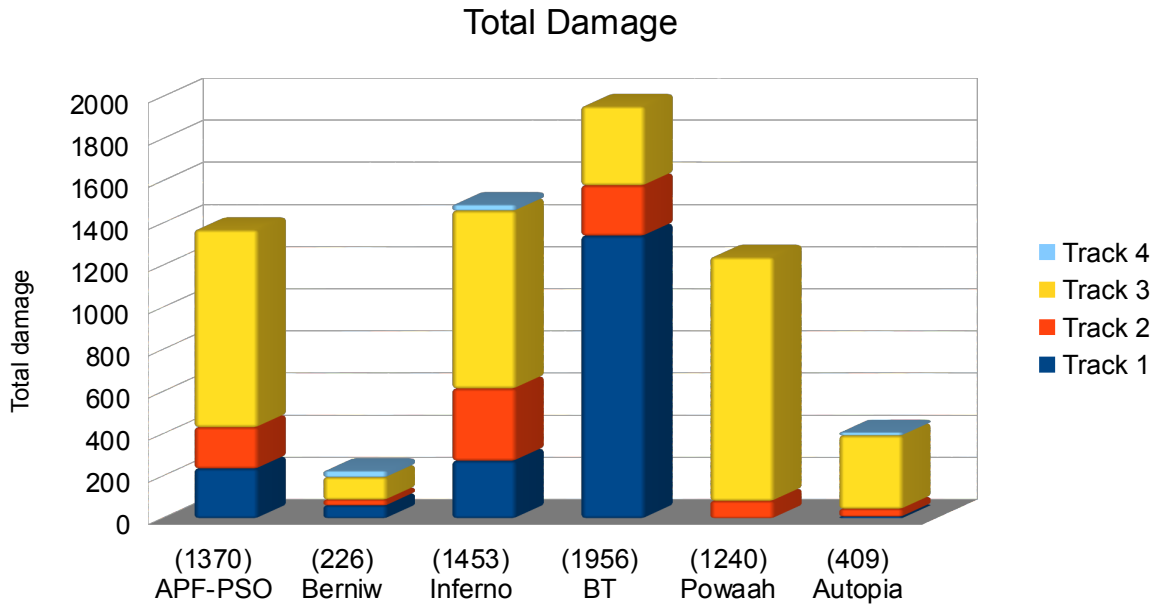


Figure 15: Total damage on all tracks

6.2 Controller Evaluation

Now we need to see the correlation of results for each controller on all tracks to get an indicator of the overall performance. This will help us in deciding which controller had a better total time, best lap time, and top speed ranking through all the races. To do this, we start by assigning points to each controller according to the F1 scoring system used in the 2011 Simulated Car Racing Championship @ CIG-2011 [3].

- 1st rank: 10 points
- 2nd rank: 8 points
- 3rd rank: 6 points
- 4th rank: 5 points
- 5th rank: 4 points
- 6th rank: 3 points

6.2.1 Experiment 1

Controller	Variable	Track 1		Track 2		Track 3		Track 4		Score	Final Ranking
		Rank	Point	Rank	Point	Rank	Point	Rank	Point		
APF	Total time	2	8	2	8	2	8	2	8	32	2
APF-PSO		1	10	1	10	1	10	1	10	40	1
APF	Best lap time	2	8	2	8	2	8	2	8	32	2
APF-PSO		1	10	1	10	1	10	1	10	40	1
APF	Top speed	2	8	2	8	2	8	2	8	32	2
APF-PSO		1	10	1	10	1	10	1	10	40	1
APF	Damage	1	10	1	10	1	10	1	10	40	1
APF-PSO		2	8	2	8	2	8	1	10	34	2

Table 6: Performance ranking for experiment 1

6.2.2 Experiment 2

Controller	Track 1		Track 2		Track 3		Track 4		Score	Ranking
	Rank	Points	Rank	Points	Rank	Points	Rank	Points		
APF-PSO	1	10	1	10	5	4	2	8	32 points	1
Berniw	2	8	4	5	1	10	4	5	28 points	2
Inferno	3	6	3	6	2	8	3	6	26 points	3
BT	4	5	2	8	3	6	6	3	22 points	4
Powaah	5	4	6	3	4	5	5	4	16 points	6
Autopia	6	3	5	4	6	3	1	10	20 points	5

Table 7: Total time ranking

Controller	Track 1		Track 2		Track 3		Track 4		Score	Ranking
	Rank	Points	Rank	Points	Rank	Points	Rank	Points		
APF-PSO	1	10	1	10	6	3	2	8	31 points	1
Berniw	2	8	2	8	1	10	4	5	31 points	1
Inferno	2	8	4	5	2	8	3	6	27 points	2
BT	3	6	3	6	3	6	5	4	22 points	4
Powaah	5	4	6	3	4	5	6	3	16 points	5
Autopia	4	5	5	4	5	4	1	10	23 points	3

Table 8: Best lap time ranking

Controller	Track 1		Track 2		Track 3		Track 4		Score	Ranking
	Rank	Point	Rank	Point	Rank	Point	Rank	Point		
APF-PSO	1	10	1	10	3	6	4	5	31 points	2
Berniw	3	6	4	5	1	10	2	8	29 points	3
Inferno	2	8	2	8	1	10	1	10	36 points	1
BT	4	5	3	6	3	6	5	4	21 points	4
Powaah	6	3	6	3	6	3	6	3	12 points	6
Autopia	5	4	5	4	5	4	2	8	20 points	5

Table 9: Top speed ranking

Controller	Track 1		Track 2		Track 3		Track 4		Total Damage	Σ Pt.	Final Ranking
	Damage	Pt.	Damage	Pt.	Damage	Pt.	Damage	Pt.			
APF-PSO	236	5	196	5	938	4	0	10	1370	24	4
Berniw	61	6	27	10	107	10	31	5	226	31	2
Inferno	273	4	343	3	844	5	29	6	1453	18	6
BT	1341	3	239	4	376	6	0	10	1956	23	5
Powaah	0	10	82	6	1158	3	0	10	1240	29	3
Autopia	8	8	36	8	349	8	16	8	409	32	1

Table 10: Total damage ranking (Pt. = Points) (Rank 1 = low damage, Rank 6 = high damage)

6.3 Discussion

We have recorded the resulting values of performance variables and represented them in the previous section to make it easier to understand and analyze them to draw our final conclusions.

6.3.1 Experiment 1

By looking at Figure 6 we can see that the APF-PSO controller has less total time on the all tracks compared to the APF controller. This gives an indicator that the use of PSO does actually enhance the performance by reducing the total time required to finish the race compared to only using APF. This also can be seen clearly by looking at Figure 7 which illustrates the accumulative total time.

The next Figure 8 shows the best lap time which every controller achieved during the race on each one of the tracks. Again it can be noticed that the APF-PSO controller has better minimum lap times than the APF controller, which is another indicator of the ability of PSO in enhancing the performance.

The maximum top speed is yet another performance indicator for both controller, we found that the use of PSO has clearly increased the speed compared to using APF only, as can be seen in Figure 9.

Finally, the total damage suffered during the race is the final performance factor that needs to be looked at more closely, since we noticed that unlike all the other performance variables, the use of PSO did not make things better in terms of the total damage, it actually made it worse. The APF-PSO controller gained more than 200% damage in all races than the APF controller, as shown in Figure 10.

By looking at Table 6 we can see that the final ranking of the APF-PSO is higher than APF in all performance variables except damage. This high level of damage might be due to the fact that when the speed is increased, the chances of track departure while turning is higher, which makes the APF-PSO controller at higher risks of colliding with the side of the track, resulting in more damage. Also, any collision at a higher speed results in far more serious damage than collisions at lower speeds.

6.3.2 Experiment 2

The second experiment was conducted by racing our APF-PSO controller against other controllers. Figure 11 shows a comparison between the total racing time for each controller on different tracks. Our APF-PSO controller seems to score good results, it is ranking first on track 1 & 2, second on track 4, and 5th on track 3. however, by looking at this figure it does not tell us much about the differences between controllers because the results are very close in most of the cases. Table 7 shows that APF-PSO got the best total time ranking of all controllers, we can consider this as a strong indicator of good performance of our APF-PSO controller.

Figure 12 gives a clear picture of the accumulative total time for each controller on all tracks, this is an indicator of the level of consistency in the controller's performance, because you can score very well on one track and very bad on all the others, so the accumulative total time for all tracks gives a good indicator on how consistent is the performance on all tracks. By examining the results in Figure 12, we can see that Berniw achieved the best accumulative total time score, our APF-PSO controller scored 3rd in the accumulative score although it was first in the ranking as shown in Table 7. This is due to the racing results on track 3, which our controller could only achieve the 5th place with a large time difference from the first 4 controllers, which added a high penalty to the accumulative total time. However, since we observed a performance drop on track 3 for all controllers, we can treat the results of this track as an outlier, which will make our controller the best in overall accumulative total time results.

The next performance indicator is the best lap time, by looking at Figure 13 we can see that the results are very close in most of the tracks, so to get more understanding of these results we have ranked each controller based on their score for each track as can be seen in Table 8. From Table 8 we see that APF-PSO and Berniw were the best controllers in maintaining a good lap time throughout the experiment. When a controller achieves a good lap time this means that it managed to keep a high speed throughout the lap, to achieve this the controller has to start the lap with high speed and it has to maintain the speed as long as possible, a controller that can achieve good lap times is a controller that can maintain high speed longer than others, which is done by following the optimal racing line.

One thing to notice is the relationship of the length of the track and the variations between different controllers best lap times, the longer is the track the more variation there is in the results. This can be seen by looking at Figure 13, we notice that track 1 & 4 the shortest tracks have very close results in best lap time, while track 3 the longest track has more variations than all other tracks. We learn from this that the track length, shape, and status are very important in determining the performance of the controller, and the more you travel on a track the more you can distinguish the differences between different controllers.

The third performance variable is top speed, by looking at Figure 14, we can see a great variation in the top speeds achieved by various controllers. However, there is some consistency in the ranking of each controller on different tracks. As we can see in Table 9, the APF-PSO controller ranked second in top speed throughout the experiment. This is a good indicator of the aggressiveness of our controller during the race, and it can also be used to explain the high level of damage suffered on some tracks. High speed can be a two edged sword, since it increases the risk of collisions and track departures, not only that, any collision at a higher speed will cause a greater damage than collisions at lower speeds.

Total damage is an important performance variable that tells us how well the controller can avoid collisions and stay on track. There are two reasons for damage, collisions and track departures. The

risk in gaining a lot of damage is that the TORCS server could shut down the controller before the end of the race. By looking at the results in Figure 15, we can see that Track 3 caused most of the damage to all controllers, this tells us that this damage is due to the shape and the muddy surface of the track which led to collisions and track departures.

The APF-PSO controller did not perform very well compared to the other controllers, it got a high level of damage and scored the 4th in damage level as we can see in Table 10, with rank 1 being the best and rank 6 being the worst in damage. The TORCS server does not tell us exactly whether the damage is caused by colliding with opponents or by track departures, but by watching the races we could see that damage mainly came from track departures and collisions with obstacles on the side of the track. But it is hard to generalize this since our human view does not accurately register all incidents for all controllers simultaneously on the track. High damage is a big performance issue that can be fixed by reducing the speed while turning or overtaking.

A deeper look into the relations between different performance factors shows that in general each track had different impact on the performance. For example, all controllers had high level of damage on track 3, and a relatively medium level of damage on tracks 1 and 2, with very little damage on track 4. We can see that performance results are both track related and controller related. Based on this, it is very important to take into consideration the type of tracks on which the controller is going to race when designing the controller.

6.4 Conclusions

The previous discussion of the experiment results showed that our APF-PSO controller achieved good ranking in most of the performance variables. The overall performance seems promising and the idea of using PSO is showing good potential according to the results.

The APF-PSO controller achieved good speed and racing time in comparison to the APF controller and to all other controllers including an APF based controller (i.e. Powahh) in the second experiment. Based on this we can state as an answer to RQ 1.1 and RQ 1.2 that the use of PSO did actually enhance the speed and reduce the racing time.

The draw back of the enhanced speed was increasing the level of damage, therefore we can say as an answer to RQ 1.3 that the use of PSO did not reduce the damage, in fact it increased the risk of track departures and collisions which at high speeds results in more damage than in lower speeds.

As a general conclusion we can state that using PSO with APF has a very good potential in building a competitive controller based on the fact that it increases the speed and reduces the racing time, however there is a major drawback to using the PSO which is the increased damage resulting mainly from lane departures which poses a threat of terminating the controller earlier in the race if it is too high in damage.

However, we can see that the damage level was not that high to result in a complete shut down of the controller, adding to the previous conclusion of enhanced racing time and racing speed we can conclude that the use of PSO does actually enhance the overall performance of a car racing controller, this can be considered as an answer to RQ 1.

6.5 Validity Threats

The first issue with validity is the limited number of tracks and opponents we have used in our experiment. We have tried to use different types of tracks ranging from wide to narrow, and straight to curvy, however, there are endless possibilities of how a racing track could look like which might affect the results dramatically. We tried to make our selection as wide and representative as possible by choosing tracks that included wide, narrow, muddy surface, and curvy. We had 5 opponents in our experiments, these opponents have a defined behavior. Changing the group of opponents could result in a drastic change in performance results and in the ranking. Therefore we have chosen our opponents to be highly competitive.

Calculation time is another factor that could be a possible cause of problems. Each game frame takes 0.2 seconds, and the server must receive a new control string within this time limit in order to execute it in the next frame. Since we are doing a lot of calculations in a very limited time we needed to make sure that we could actually pass the desired control string within time before the next game frame, otherwise the resulting delay could result in executing the wrong control instruction at a given time which results in unwanted car behavior. To face this problem we have enabled the time limit constraint in the server, which means that the server will shut down the controller if it does not pass a new control string every 0.2 seconds.

Another issue with the validity of our conclusions is determining the source of damage whether it is caused by collisions with other opponents or by track departures. Since we could not get any information from the server on the source of damage suffered by our controller, we watched the race by eye and identified that most of the damage came from track departure not from colliding with other opponents.

6.6 Future Work

Our work has a great potential to be improved and to be the basis of even better solutions to build autonomous car controllers. This is due to the flexibility of the APF and PSO algorithms which can be modified in many ways.

Based on the current results it is clear that there is an issue with managing high performance and keeping the level of damage at lower levels at the same time. This could be the base for a future work where training the particles at higher speeds to find optimal solutions within a safety range to prevent track departures and collisions which are the main sources of damage.

Another aspect is to use learning with particles. Particles can be trained using Machine Learning or Optimization techniques to reduce the search space to more promising areas, and reduce the convergence time.

Taking the track surface with consideration by increasing or decreasing the turning and overtaking speed, since this was a major problem on track 3 which had a slightly muddy surface and resulted in a high level of damage for all controllers.

Finally, controlling the convergence speed of particles is yet another interesting feature that is worth looking into, the aim should be reducing the overhead and speeding up the decision process.

REFERENCES

- [1] C. Athanasiadis, D. Galanopoulos, and A. Tefas, "Progressive Neural Network Training For The Open Racing Car Simulator," in *IEEE Conference on Computational Intelligence and Games (CIG'12)*, 2012.
- [2] L. Cardamone, D. Loiacono, P. L. Lanzi, and A. P. Bardelli, "Searching for the Optimal Racing Line Using Genetic Algorithms," presented at the IEEE Conference on Computational Intelligence and Games (CIG'10), Dublin, 2010, pp. 388 – 394.
- [3] D. Loiacono, L. Cardamone, M. Butz, and P. L. Lanzi, "The 2011 Simulated Car Racing Championship @ CIG-2011," in <http://www.slideshare.net/cardamoneluigi/the-2011-simulated-car-racing-championship-cig2011>, Seoul, Korea, August 31-September 3.
- [4] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, 1991, pp. 1398–1404.
- [5] J. Kennedy and R. Eberhart, "Particle swarm optimization in Neural Networks." 1995. Proceedings., IEEE International Conference on Neural Networks, volume 4, pages 1942–1948 vol.4.
- [6] M. BEDNÁR, A. BRČEK, M. BRIS, M. FLOREK, V. JUHÁSZ, J. KOSMEL, and I. VALENČÍK, "The Modular Architecture of an Autonomous Vehicle Controller," presented at the Slovak University of Technology, Faculty of Informatics and Information Technologies Ilkovičova 3, 842 16 Bratislava, Slovakia.
- [7] J. Hagelbäck and S. J. Johansson, "The Rise of Potential Fields in Real Time Strategy Bots," Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, pp. 42–47, 2007.
- [8] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, 1986.
- [9] M. T. Wolf and J. W. Burdick, "Artificial Potential Functions for Highway Driving with Collision Avoidance," presented at the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 2008.
- [10] P. Vadakkepat, "Evolutionary artificial potential fields and their application in real time robot path planning," *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, pp. 256–263.
- [11] R. Poli, J. Kennedy, and T. Blackwell, "Particle Swarm Optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.
- [12] R. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII*, 1998, pp. 611–616.
- [13] S. Gheitanichi, F. Ali, and E. Stipidis, "Trained Particle Swarm Optimization for Ad-Hoc Collaborative Computing Networks," in *AISB 2008 Convention Communication, Interaction and Social Intelligence*, 2008, vol. 1, p. 7.
- [14] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Simulated Car Racing Championship Competition Software Manual," presented at the Politecnico di Milano, Dipartimento di Elettronica e Informazione, Italy, 2011.
- [15] S. S. Ge and Y. J. Cui, "New Potential Functions for Mobile Robot Path Planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, Oct. 2000.
- [16] S. S. Ge and Y. J. Cui, "Dynamic Motion Planning for Mobile Robots Using Potential Field Method," *Autonomous Robots, Kluwer Academic Publishers.*, no. 13, pp. 207–222, 2002.
- [17] X. GU, X. Wang, and W. Li, "Optimization of Trajectories Based on APF-PSO with Radar Threats," presented at the Signal Processing, Communications and Computing (ICSPCC), 2011

- IEEE International Conference, Xi'an, China, 2011, pp. 1 – 4.
- [18] S. Hassan and J. Yoon, "Smart Targeted Drug Delivery System Using Swarm Based Drug Loaded Nano Particles Simulation," presented at the 4th International Conference on Intelligent Systems, Modelling and Simulation, Bangkok, Thailand, 2013.
 - [19] A. Agapitos, M. O'Neill, A. Brabazon, and T. Theodoridis, "Learning Environment Models in Car Racing using Stateful Genetic Programming," presented at the 2011 IEEE Conference on Computational Intelligence and Games (CIG), Seoul, Korea, 2011, pp. 219–220.
 - [20] T. Uusitalo and S. J. Johansson, "A Reactive Mutli-agent Approach to Car Driving using Artificial Potential Fields," presented at the IEEE Conference on Computational Intelligence and Games (CIG), Seoul, Korea, 2011, pp. 203–210.
 - [21] A. P. Engelbrecht, *Computational Intelligence, An Introduction*, 2nd ed. West Sussex, England: John Wiley & Sons, Ltd, 2007.
 - [22] E. Onieva, D. A. Pelta, J. Godoy, V. Milanes, and J. Perez, "An Evolutionary Tuned Driving System for Virtual Car Racing Games: The AUTOPIA Driver," *International Journal of Intelligent Systems*, no. 27, pp. 217–241, Mar. 2012.