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.

Master's Thesis
Computer Science
Thesis no: MCS-2013-10
Month Year

Collision Detection and Overtaking Using Artificial Potential Fields in Car Racing game TORCS using Multi-Agent based Architecture

Muhammad Salman
musc10@bth.se
salmanikram2003@gmail.com

Contact Information:
Author: Muhammad Salman
Address: Lindblomsvägen 98 LGH 1205 , 372 33 Ronneby Sweden
E-mail: salmanikram2003@gmail.com

University advisor(s):
Dr. Stefan J. Johansson, PhD
School of Computing
Blekinge Institute of Technology

School of Computing
Blekinge Institute of Technology
SE – 371 79 Karlskrona
Sweden

Internet : www.bth.se/com
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

The Car Racing competition platform is used for evaluating different car control solutions under competitive conditions [1]. These competitions are organized as part of the IEEE Congress on Evolutionary Computation (CEC) and Computational Intelligence and Games Sym-posium (CIG). The goal is to learn and develop a controller for a car in the TORCS open source racing game [2]. Oussama Khatib [3] (1986) introduced Artificial potential fields (APFs) for the first time while looking for new ways to avoid obstacles for manipulators and mobile robots in real time. In car racing games a novel combination of artificial potential fields as the major control paradigm for car controls in a multi-agent system is being used to coordinate control interests in different scenarios [1]. Here we extend the work of Uusitalo and Stefan J. Johansson by introducing effective collision detection, overtaking maneuvers, run time evaluation and detailed analysis of the track using the concept of multi-agent artificial potential fields MAPFs. The results of our extended car controller in terms of lap time, number of damages and position in the race is improved.

Keywords: Artificial Potential Fields, TORCS, Simulated Car Racing Championship, Collision Detection, Overtaking

ACKNOWLEDGMENTS

"In the name of God Almighty, the Beneficent, the Merciful"

I would like to thank my supervisor Dr. Stefan J. Johansson, who was very kind and supportive during my thesis work. He has been a great help and support on the way to my findings. I will always look forward to work under his kind supervision in the future.

I would also like to thank my parents for their prayers and support during my long stay outside the country.

I would also like to thank the developers behind the open racing car simulator (TORCS) for providing such a wonderful platform to test AI concepts and techniques., the star up tutorial written by Bernard Way man to develop a controller bot in TORCS.

The last acknowledgement goes to the organizers of the simulated car racing championship. They developed a client server based module to TORCS in a rather complex but challenging environment.

Contents

LIST OF TABLES

TABLE OF FIGURES

1 INTRODUCTION

In every car racing game, the aim of the players involved is to finish the race first. Similarly in TORCS [2], the AI bots compete for the first position in the race. The racing involves number of difficulties to overcome like keeping the car on track, avoid collision with the track boundaries as well as opponents and to minimize the lap time as much as possible.

1.1 TORCS

Open Racing Car Simulator (TORCS) is an open source car racing simulator with 3D features. It provides a fully customizable environment that can be used by researchers in the field of computational intelligence for the purpose of benchmarking. Eric Espié and Christophe Guionneau originally created TORCs, but Bernhard Wymann is now the head of the project development team. It is written in C++ programming language and distributed under the GNU GPL license. Pre programmed AI car controllers (bots) are developed for TORCS that race against each other while the users are provided a car to control via joystick, keyboard or mouse [2]. One among the dominant features of The TORCs is the state of the art physics engine that makes it possible to handle the collision, fraction of the wheels, aerodynamics and other important aspects of the racing car. TORCs employs a modular approach and the controllers are implemented as separated software modules which make it easy to develop a controller and integrate it with the game server [4][5].

Figure 1: Our Car Controller in Simulated Car Racing Game TORCS.

1.2 Simulated Car Racing Championship

Several Competitions are held at major conferences in the field of Evolutionary Computation, Computational Intelligence and games. Simulated Car Racing Championship is one among the many held at different occasions in those conferences. The aim of the competition is to design a controller that makes use of the concepts use in the field of Evolutionary computation and computational intelligence. Each controller is provided a practice session on unknown tracks without opponents. The purpose of the practice session is to make the controller aware of the track it will compete with other drivers in the real competition race. A number of sensors are provided to the controllers in order to perceive the racing environment. Each sensor provides the relevant information regarding the car surroundings (e.g. the track edges and boundaries, the position of the different segments of the track), of the car situation (e.g., current fuel, the engine RPMs, the current speed, the current gear, etc.), and updated information regarding game state such as lap time, position in the race etc. The controller perform a number of typical driving actions such as to steer the wheels, break, gear changing and adjusting clutch values etc [6].
We based our findings on the TORCS server as well as client server competition software [6]. For the purpose we developed two controllers that test the same concept in different scenario. The TORCS server provides complete information about the tracks, own car, opponents car and a number of server's configuration parameters. The competition software is client server based and provides access to sensor based information regarding cars, opponents and tracks to the controller. Some of the sensors and effectors with description are depicted in Table 1.

Name	Range (unit)	Description
accel	[0,1]	
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	From -1 to 7	Gear value.
focus	[-90,90]	Vector of 5 range finder sensors: each sensor returns the distance between the track edge and the car within a range of 200 meters.
angle	[-π, +π](rad)	Angle between the car direction and the direction of the track axis.
steering	[-1,1]	Steering value: -1 and +1 means respectively full right and left, that corresponds to an angle of 0.785398 rad.

Table 1: Some of the available sensors and effectors

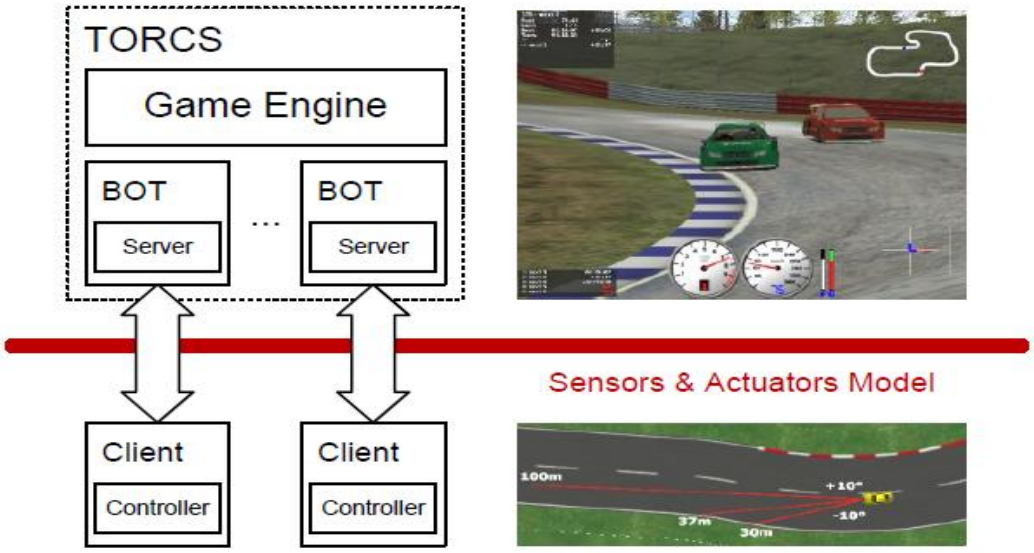


Figure 2: The architecture of the competition software.

The client receives sensor data i.e. car control data from the server and decide values according to the designed strategy for the controller bot. the information is then passed back to TORCS server to drive the controller via calculated values. The sensors are described here in details.

Track sensors: Track sensors represent a vector of 19 range finders. Each sensor in the vector collection returns the distance between track edge and the car. The distance is returned if the car is within range of 200 meters from the track edge. The sensors scan the space spanning clockwise in the range from $+\pi/2$ to $-\pi/2$ in front of the car. The scanning is performed in front of the car every 10 degree.

Opponent sensors: Opponent sensors represent a vector of 36 opponent sensors. Each sensor returns the distance to nearby opponent in the span of $\pi/18$ every 10 degree within specified range of 200 meters.

Track position: Track Position sensor returns distance between the car and track axis. It returns three values, could be 0, 1 or -1. The 0 return value corresponds to the middle of the track while -1 and 1 indicates the car position on right or left side of the track respectively.

Track angle: It returns the angle between the car direction and direction of the track axis. The angle is always in radians.
Gear: Provides what gear we currently are in

SpeedX: It returns speed of the car in longitudinal axis.

Distance from start: This sensor returns the distance travelled by the car along the track from the start line.
We mentioned here some of the sensor data but there are a number of other sensors available which are briefly described in the competition software manual [6].

1.3 Motivation

Several simulated car racing competitions have been organized in the last five years. These competitions were organized in conjunction with the leading international conferences. Researchers were told to submit their entries in the form of AI car controllers for a racing game. These controllers raced against each other on available tracks and the one with best lap time, less number of damages and 1st position in the race was considered the winner. IEEE CEC and the IEEE CIG in particular organize a group of competitions which are based on popular board games (i.e. *Go* and *Othello), video* games (such as *Pac-Man*, *Super Mario Bros*, and *Unreal Tournament) and car racing games (such as TORCS)*.
The dynamic environment of car racing games like TORCS provides a perfect platform to test computational intelligence and artificial intelligence concepts. Collision detection and overtaking in these games are considered the complex and technically demanding tasks in the fast changing game environment. Traditional techniques like boundary intersection and others has long being used for collision detection in games. The Artificial potential fields emerges as a new technique for real time obstacle avoidance when first used by Oussama Khatib [3]. Hagelbäck and Stefan J. Johansson [7] applied the same concept successfully to open real time strategy (ORTS) games. Later its use is being extended to open racing car simulator (TORCS) [1]. In the previously mentioned research work artificial potential fields were successfully implemented for the purpose of path finding and to avoid collision with the track edges. Here in this thesis we use multi agent based artificial potential fields by defining different strategies and techniques to handle opponents so that we can compete for 1st position in the race.

1.4 Thesis structure

We divided the thesis work into different chapters each of which reflects its different contribution towards the thesis work. The chapters and their brief description are stated as:

- Chapter 2: the related work that has been done in the field of this thesis.
- Chapter 3: Research Questions and Hypothesis, how the questions are addressed and hypothesis being verified.
- Chapter 4: the methodology being used to address the research questions and approaches used to achieve the goal of this thesis.
- Chapter 5: Strategies and techniques to solve the research problem are discussed and elaborated.
- Chapter 6: the experiments and their results.
- Chapter 7: Discussion
- Chapter 8: Conclusion and Future Work

2 RELATED WORK

The Car Racing competition platform is used for evaluating different car control solutions under competitive conditions [1]. These competitions are organized as part of the IEEE Congress on Evolutionary Computation (CEC) and Computational Intelligence and Games Symposium (CIG). The goal is to learn and develop a controller for a car in the TORCS open source racing game [8].

TORCS, the open Racing Car Simulator is an open source 3D car racing simulator which is available for the existing operating systems. It is a portable multi-platform car racing simulation first developed by Eric ESpie and Christophe Guionneau, but project development is now headed by Bernhard Wymann. The simulator is developed in C++. Pre programmed AI drivers are developed and integrated with the TORCS to race against each other while enabling the user to control a vehicle using keyboard, mouse or other input devices appropriate for the game [2][4].

Recently several articles have been published which focus on car racing games. The techniques applied so far cover almost the entire computational Intelligence field and neural networks. Programming natural and believable behaviors in games is increasingly challenging in modern games. It is eminent from the recent works in the literature related to AI that computational intelligence can provide an effective solutions to complex game behavioral problems and non player characteristics (NPC) [9][10][11].

Development of Learning robots or humanoid robots is long being the topic of interest in the AI community. This is usually aimed at replacing non player characteristics (NPCs) into more realistic and human like characters. Set of rules were first used by Priesterjahn [12] in order to develop a NPC for the game Quake III. He collected data from human player and derived the set of rules which were then optimized by using evolutionary algorithm [5]. The combination of learning algorithms has also been used such as combining reinforcement learning, Bayesian motion modeling and fuzzy clustering in order to develop an NPC for the game Quake II [13].

Artificial Potential Fields was first introduced by Oussama Khatib [3] in 1986 while he was investigating, the methods and techniques for real time obstacles avoidance in case of manipulators and mobile robots. The main idea was to move the manipulator in a field of forces comprises of attractive as well as repulsive poles. The attractive poles are used to achieve various goal positions while repulsive poles are mainly used to avoid collisions with the associated obstacles.

Rossetter [14]applied the artificial potential fields to control a vehicle while he was looking for the new approaches to prevent accidents from lane departures. He presented a framework for lane keeping assistant based on artificial potential fields and provided mathematical bounds on performance of lane keeping for the system, which can be used to design a potential field controller capable of lane keeping in close coordination with the human driver.

Thurau et al. [13] used the concept of APFs (Artificial Potential Fields) in a 3D first person shooter (FPS) game called Quake II. They developed a bot that learn human like reactive behaviors provided by the available training sets for the game. In their work, they used artificial potential fields primarily for the purpose of path finding and navigation due to its computationally and goal oriented control of movement.

3 RESEARCH QUESTION AND HYPOTHESIS

In this section we will introduce the research question, followed by the hypothesis and finally how the hypothesis is verified.

3.1 Research Questions

RQ1. How we can best use the advantages of APFs (Artificial Potential Fields) for achieving a successful overtaking behavior in a car controller for TORCS?

RQ2. How to use APFs for collision detection among the dynamic objects like opponents in a car racing game?

RQ3. How to compare and evaluate the overtaking performance of the bots in the simulated TORCS environment?

3.2 Aims and Objectives

We will extend the previous research work [1] by introducing the overtaking behavior to the developed car controller using the same concepts of Artificial Potential Fields. The collision detection was not properly implemented that contribute to bad implementation of overtaking behavior in the previous work. The aim is to make an effective use of APFs in order to achieve a successful overtaking behavior for car controller in TORCS. We will scan the track for detailed analysis including finding the radius of various segments of the track which is later on used for speed control during curves. Sub tasks in the research are as follow:-

- Path finding
- Track Recognition
- Overtaking Maneuvers
- Effective synchronization amongst the above mentioned tasks

3.3 Hypothesis

In view of our research questions described in section 2.1, we formulate our hypothesis as

"**Does a Multi-agent based APFs solution provide better results than the traditionally used methods for collision detection and overtaking in a car racing games**?"

In order to apply APFs in car racing game, we require a number of different potential fields that corresponds to different objects of interest in the game world like track segments (left Curve, Right Curve, Straight), own car and opponent cars etc. To develop a competitive controller, we need to drive as fast as possible along the track without losing the car control as well as to avoid collision with the opponents and track boundaries. To keep things simple, we approach the problem with the following modular divisions.

3.3.1 Path Finding and Navigation

To drive the car in track direction in a car racing game, we need to place attractive as well as repulsive fields in order to navigate along the track. Attractive fields are placed along the shortest path possible to achieve the goal positions while repulsive fields are placed along the track boundaries to avoid collision.

3.3.2 Fields of Curvature

An important aspect of car racing is to maximize the speed while driving along the curves on the road. At high speed the car may lose control if properly not handled. For this purpose the curvature is minimized to achieve the maximum speed possible along the curves. We use field of curvature for this purpose.

3.3.3 Collision Detection

Collision detection is always an important aspect of every car racing game. Our AI car controller must be able to avoid collision with opponents competing in the same race. For the said purpose we place repulsive charges along the side of approaching opponent's , when it approaches our car in range equal to a pre-defined minimum collision distance. To avoid collision with the track edges the forces that keep the car on track are used described in section 3.3.5

3.3.4 Overtaking

Successful overtaking strategies play a major role in winning a car racing game. We place attractive forces along the track for our controller to reach that position. As the overtaking strategies modify the target points therefore the corresponding fields are altered. This phenomenon creates a new set of artificial potential fields which are referred as "Field of Opponent (FOO)" fields.

3.3.5 Keep the car on track

To drive in the track direction without any collision with the track edges, we must have repelling as well as attractive forces spread all over the track to keep the car on track. The repelling forces are concentrated on the edges of the track while the attractive forces are spread along the preferred navigational path. For example if our controller wants to follow the middle of the track throughout the race, the attractive forces are placed in the middle of the track along the length of the whole track.

3.4 Verification of Hypothesis

The hypothesis will be verified on two platforms i.e. TORCS server [2] and simulated car racing championship software [6] . The TORCS server provides complete information about the tracks, own car, opponent cars and car physics which could be used to get better performance than the simulated car racing championship software. On the other hand simulated car racing championship software is based on TORCS server but provides limited information about the game environment. The TORCS server still remains the test bench for the experiment even if the simulated car racing championship software is used.

4 METHODOLOGY

The open racing car simulator (TORCS) provides a rich and complex game environment with 3D features. Therefore we choose to adopt a modular approach to address our research problem. For this purpose we have selected Multi-Agent Potential Fields (MAPFs). Multi-Agent Potential Fields were first introduced and applied by Johan Hagelbäck and Stefan J. Johansson [7] in real time strategy games. They applied artificial potential fields for path finding and collision detection in order to avoid obstacles.

4.1 Artificial Potential Fields

The core concept of the MAPFs based solution is artificial potential fields (APFs). Hagelbäck and Stefan J. Johansson [7] identified three types of potential fields in their work i.e. Field of Navigation, Strategic Field, and Tactical field. They identified objects of interest in game world for open real time strategy (ORTS) game and constitute associated potential fields accordingly. The static terrain in the game generates field of navigation with repelling force. The purpose was to avoid obstacles and detect collision. Strategic field, they defined was attractive in its nature and scope and was used to enable agents navigate to the opponents while keeping a safe distance to fight. Small repelling fields were also used to avoid collision with the friendly units, own bases and sheep. Some of the fields they generated and their description are depicted in the Figure 3. In the figure, E is an attractive charge with highest potential while the mountains (brown), own and enemy units (green and white dots) have repelling charges around them.

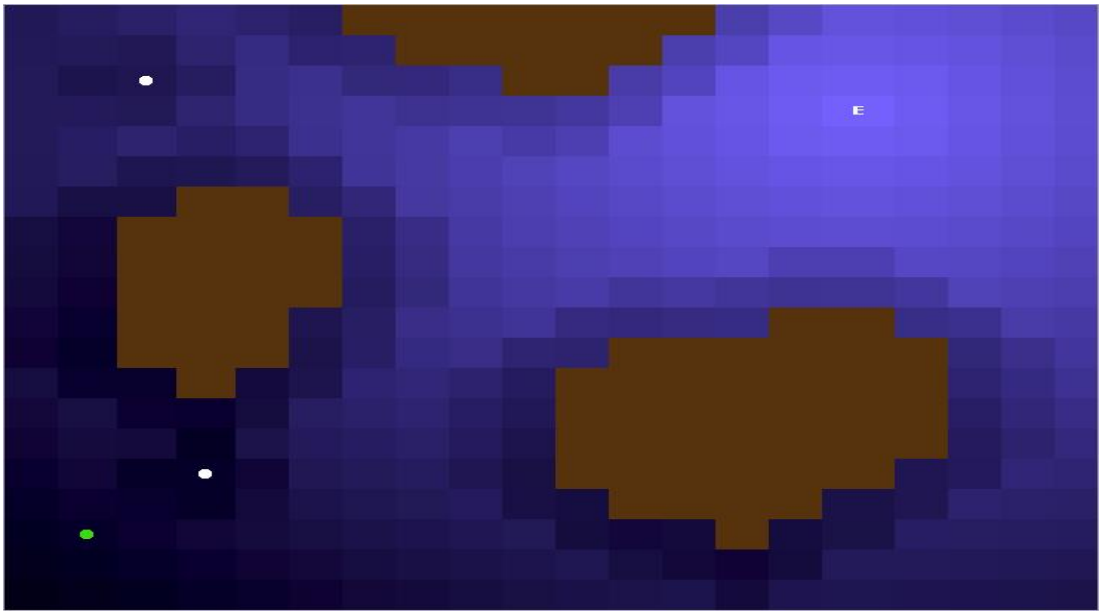


Figure 3: Illustrates the charges and associated fields. The dark blue tiles represent low potential area while the lighter one represents high potential area, Figure by Hagelbäck [7]

Moving a unit from its present location to the target point E in the field of forces spread all over the game world is achieved using lookahead positions. The lookahead position pointing to the closest title is considered. If the potential of that tile is higher than the other titles in the game region, the unit is moved there. The process continues until the unit reaches its target point located at E. This is illustrated in Figure 4.

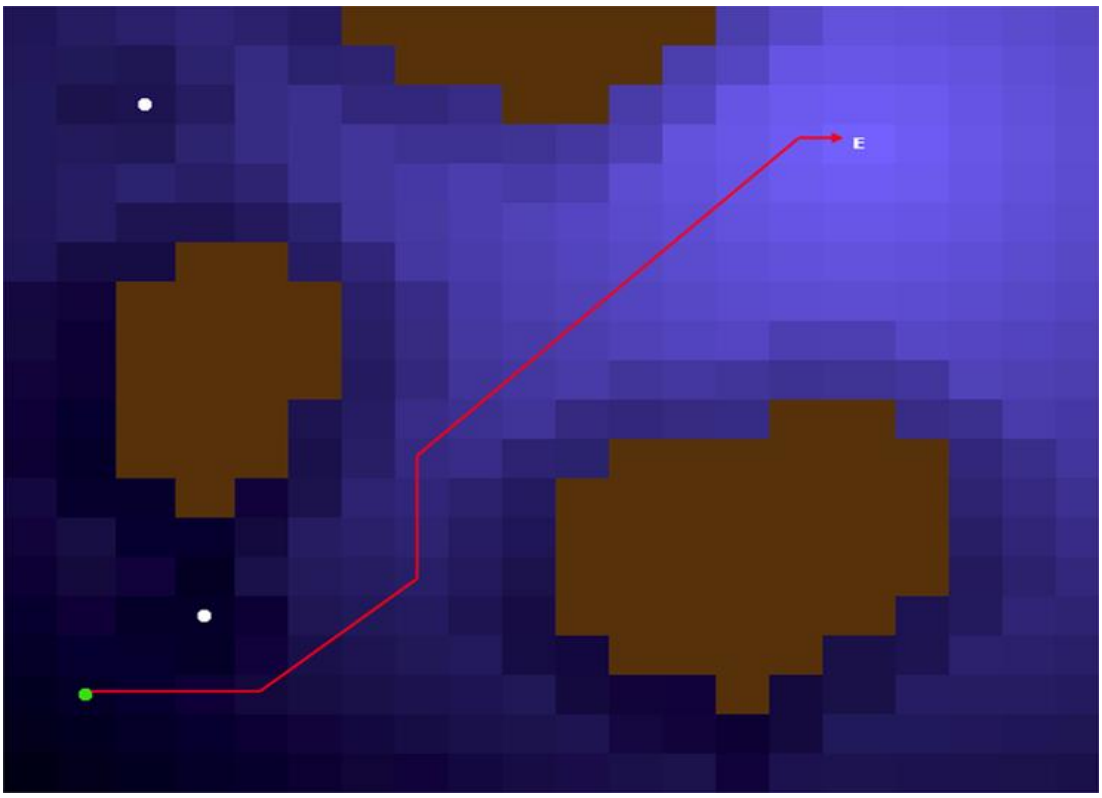


Figure 4: Illustrates the path navigation in the field of forces. The green unit moves from its current position to the target point located at E avoiding other units and obstacles (mountains), Figure by Hagelbäck [7].

4.2 Multi-Agent Potential Fields

Hagelbäck and Stefan J. Johansson [7] identified six phases in the design for a Multi-Agent Artificial Potential Field (MAPF) based solution. As the open racing car simulator (TORCS) game environment is similarly complex like the open real time strategy (ORTS) games, therefore we have chosen similar MAPF based methodology for our solution. The design steps are as follow:

1. The identification of objects
2. The identification of the driving forces (fields) of the game
3. The process of assigning charges to the objects
4. The granularity of time and space in the environment
5. The agent system, and,
6. The architecture of the MAS

4.3 The identification of objects

The car racing simulator's (TORCS) environment is a collection of a large number of objects including 2D and 3D objects. We will choose only those objects that either directly or indirectly interacts with our controller. The division of objects into static and dynamic category is important because it affects the way artificial potential fields are applied. The objects of interest we identified in TORCS are:

TRACK:

Track is a static object in the car racing simulator's (TORCS) environment. The track structure is divided into three main segments i.e. Straights, Right Curve and Left Curve. A controller designed and programmed in the server has full access to track information while the one programmed for the client server based simulated car racing championship has access to limited information via track sensors provided.

Own Car:

Own Car refers to the car our programmed AI driver will control during the race. It is a dynamic object because it drives along the track and interact with other cars called opponents. The TORCS server and its client server based Simulated car Championship both provides complete information about the car states such as angle in track direction, current speed, current gear, clutch value and distance from start line etc.

Opponents:

Opponents are similar dynamic objects like Own car. Opponents are also preprogrammed AI drivers but a human controlled version is also provided that could also be used as opponent in the TORCS server. In Simulated Car Racing Championship, the information regarding opponents is accessible through opponent's sensors.

4.4 Motivating Forces in TORCs

To finish the race first is the main goal of every car racing game. Similarly TORCS and its pre-programmed AI drivers compete with each other for the first position. The purpose of the Simulated Car Racing championship is to provide a platform for AI drivers to compete with each other provided the same set of information regarding the game environment. We identified a number of driving forces rather at abstract level but help understand the overall road map towards the ultimate goal i.e. first position in the race. The driving forces are:

- Keep the car on track
- Drive along the track as fast as possible
- Avoid collision with track edges and competing opponents
- Keep the car in control while driving fast importantly during curves.

Cardamone et. al. [15] used shortest path (SP) and the minimum curvature path (MCP) with genetic algorithm to find best racing line while driving along a track in car racing game. He divided the track into different segments and applied genetic algorithm to balance the racing line between shortest path (SP) and minimum curvature path (MCP). Previously field of track (FOT), field of Shortest Path (FOSP) and field of curvature (FOC) to achieve the best

racing line is being used in car racing game [1] . We identified the fields inspired from the work of [15] and [1] . The fields along with descriptions are:

Fields of Track:

We divided the track into four segments i.e. Straight, Right Curve, Left Curve and track boundaries which creates four different fields for each segment i.e.

- **Field of Track for Straight Segments (FOTS)**
  the TORCS [2] server divides the track into three segments type i.e. straights, left curve and right curve. Therefore we assign different fields to each segment and FOTS is one among them for the straight segments of the track.
- **Field of Track for Left Curve (FOTLC):**
  It is similar to FOC in [1] but includes the detailed analysis of the track curve i.e. type of curve (short or long), its radius and offset change (target point may change due to overtaking or avoiding opponents).
- **Field of Track for Right Curve (FOTRC)**
  It is similar to FOTLC with the exception in direction of the curve.
- **Field of Track for Boundaries (FOTB)**
  In order to keep the car on track and avoid collision with the track edges, repelling charges are placed along the detected edges of the track with negative potentials. Attractive charges with positive potentials are concentrated in the middle for straight segments which changes otherwise in curves due FOTLC and FOTRC.

Field of Opponent (FOO):

In order to avoid collision with the opponents, a field of opponent (FOO) is created when our car approaches at a predefined striking distance of opponent. The opponent position is provided both by the TORCS [2] and Simulated Car Racing Championship [6] software.

4.5 Objects Assignment to different fields

The forces identified with respect to the existing fields, the strength of each field which we term weights in our scenario and the placement (position) along the track segments relative to our own car is carried out by agents. The corresponding agents and their description can be found in Section 5.3.

4.6 The granularity of time and space

Decision making in car racing games is usually time bound. The response to the fast changing game events must be robust and accurate. Artificial potential fields combined with multi agent based modular approach make it possible to respond in timely manner by taking decision every frame. In our tests we had no problem evaluating 171 lookahead points (which we have chosen to represent as a decision) in a resulting frame rate of 150 frames per second or higher. A description of our lookahead points can be found in Section 5.2.

4.7 MAS Architecture

We designed MAS Architecture for our car controller bot in a way that each agent handles a unique aspect of the car racing. By definition agents are autonomous with the ability to communicate with each other. Therefore new agents can easily be integrated into the architecture and similarly existing agents can be activated and deactivated. The agents calculate charges with associated fields and send it to TORCS server with car data such as position of the calculated charge and its potential, angle, steer, target speed etc. Figure 5 depicts the agents and the process of inter agent communication. Each agent receives sensory information from the TORCS server via interface agent which is processed in its own way. The driver agent communicates with all the agents in the system such as track agent, curvature agent , speed agent, aerodynamic agent and overtake agent to collect car control data which is processed and returned to the TORCS server via interface agent.

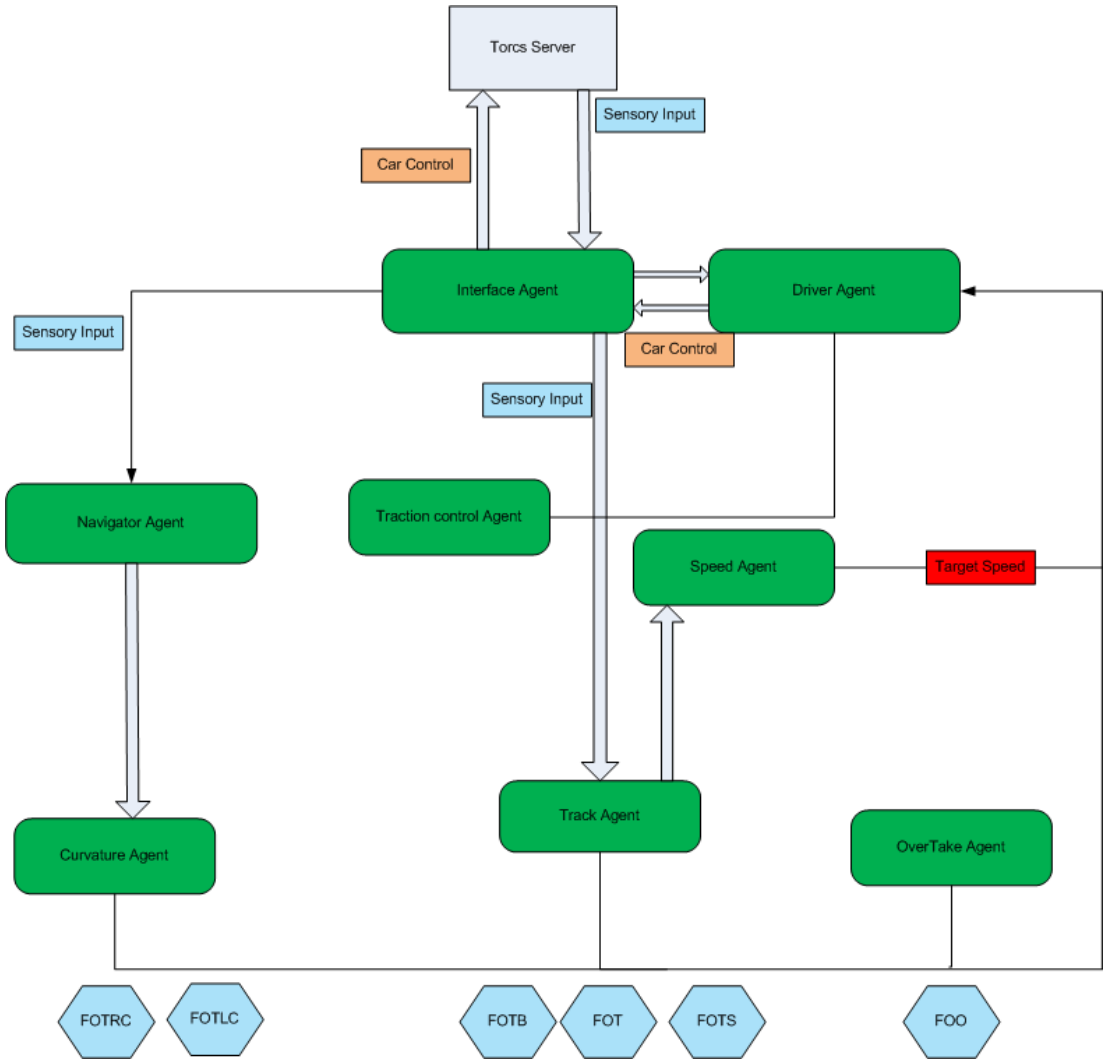


Figure 5: MAS-Architecture of our car controller. Different agents and how they interact with each other and the server is shown.

5 STRATEGIES AND TECHNIQUES

This section explains the strategies and techniques being used in the development of our car controller bot. The techniques differ in some aspects for the two versions of our car controller i.e. one developed for the TORCS server itself and the other for the Simulated Car Racing Championship client server based software. TORCS server provides direct access to track, car and opponents structures while the Simulated Car Racing Championship provides indirect access via sensors from their server module in the TORCS engine. We will explain strategies and techniques for both the versions of our controller bot in separate sections followed by the agents which are common for both the versions.

5.1 Track

5.1.1 Track View (TORCS)

We have used the robot skeleton provided by the Bernhard [16] for the TORCS version of our controller. We fetch track information from the track header which comes with the source of the TORCS server. The track is divided into three types of segments i.e. left turns, right turns and straight segments which connect tangentially to each other defined in the tTrack structure in the header. The properties differ from segment to segment and depend upon on the type of segment e.g. a straight segment has a width and length while turns have additional property of radius.

```
typedef struct
{
    const char *name;   /**< Name of the track */
    const char *author; /**< Author's name */
    char       *filename;    /**< Filename of the track description */
    void       *params;  /**< Parameters handle */
    char       *internalname; /**< Internal name of the track */
    const char *category;   /**< Category of the track */
    int        nseg;     /**< Number of segments */
    int        version;  /**< Version of the track type */
    tdble      length;   /**< main track length */
    tdble      width;    /**< main track width */
    tTrackPitInfo pits;     /**< Pits information */
    tTrackSeg     *seg;     /**< Main track */
    tTrackSurface *surfaces;    /**< Segment surface list */

    t3Dd        min;
    t3Dd        max;
    tTrackGraphicInfo   graphic;
} tTrack;
```

Figure 6: tTrack structure in the Track header file.

5.1.2 Track Sensors (Simulated Car Racing Championship)

Simulated Car Racing Championship [6] software provides a client with sample controller which make use of track sensors in the degrees between -20 to 20 after

every 5 degrees spanning from [-90,-20] degrees and [20,90] after every 15 degrees. These default sensor settings focus on the scenario in front of the car rather than considering the sides. We maintain these default settings for our car controller bot.

5.2 Lookahead Points

Our approach in using Lookahead points is different than the one used by Hagelbäck and Stefan J. Johansson [7]. In their work, they represented each tile in the game world as a potential in the field which form the basis of their path finding and obstacle avoidance methodology. As we have different scenario both in terms of platform used and the information available from the game world, our methodology for using look ahead points differ than the one they used. Our Lookahead (LA) points grow with the speed of the car in track direction as the case with the normal driving. In the version for TORCS server, we follow the track middle lookahead meters. The formula which calculates lookahead looks like

$$L = \infty + s + \beta$$

Where

- L means lookahead point,
- ∞ means lookahead constant,
- s means speed of the car in track direction and
- β mean lookahead factor

As mentioned earlier that the track is divided into segments, therefore the length of lookahead distance is considered to find the segment where our target point is located. Target points along with positions on the segments and associated potentials depends upon the type of segment they are placed in e.g. location of the target point on the straight segment remains at the middle while in turns, they are placed at the opposite side of the curve to minimize curvature and achieve shortest path as well to maximize speed. The agents described in the section 5.3 are responsible to calculate different fields, their positions and resulting angle, brake and throttle etc.
In Simulated Car Racing Championship software, we don't have access to exact track information, rather the distance to track edges is measured every frame and relayed to the controller via sensors. In this scenario we used lookahead points as a control paradigm in the same way Uusitalo and Stefan J. Johansson [1] used it in their work. An action in this regard corresponds to the angle of car, brake value, throttle value and wheel position in the current frame or game tick. The lookahead points are assigned potentials according to the corresponding fields and the one with highest potential is chosen for the action in the current frame. The placement (position) of these lookahead points takes into account a number of inputs related to the car controlled by our bot such as

- The angle of the car in track direction.
- Velocity of the car.
- Current segment of the track.

5.2.1 Determine the length of lookahead points

The lookahead distance as we described in section 5.2 is computed with the current speed and grows with the increasing speed likewise. The distance raced and distance from the start line sensors provided by the competition software first contribute to the initial length measurement. The formula to determine the length of the lookahead points is:

$$L = d_{distToSegEnd} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots . eq(1)$$

Where

- L= length
- $d_{distToSegEnd}$ = distance to the end of current track segment

Distance to the end of the current track segment is calculated separately for straight segments and curves. The formulas are:

For Straight segments

$$d_{straight} = l_{currtSegLength} - d_{FromStartLines} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots eq(2)$$

Where

- $d_{straight}$ = distance to the end of straight segment.
- $d_{FromStartLines}$ = distance provided by "distance from start line" sensor.
- $l_{currtSegLength}$ = length of the current track segment calculated from current track position.

For Turns

$$d_{curve} = (TrkPos_{arc} - d_{FromStartLine}) * TrkPos_{rad} \ldots\ldots\ldots\ldots eq(3)$$

Where

- d_{curve} = distance to the end of the turn.
- $TrkPos_{arc}$ = arc of the turn.
- $TrkPos_{rad}$ = radius of the turn.
- $d_{FromStartLines}$ = distance provided by "distance from start line" sensor.

5.3 The Agents

Here we will describe how we have applied the charges in the fields, currently letting the agents being responsible for that.

5.3.1 Driver Agent

The Driver Agent performs the basic functions of car driving. It performs a bridging role between the interface agent and other agents of the system. It provides relevant inputs to other agents regarding game environment such as tracks, own car (current speed, track position, position in the race, number of damages etc), and lookahead points. It updates the game state every frame and relay updated information to track agent, curvature agent, speed agent, overtaking agent and aerodynamic agent which process the inputs and return the results. How the agents process these information such as calculating charges and their potentials, steering angles, brake and gear values and allowed speed etc is described in the relevant agent section. We divided the tasks performed by the Driver Agent in to the following categories.

- **Track Initialization**
  Initialize the track for the race.
- **Driving**
  Drive the car in the track direction by adjusting steer, acceleration, clutch, brakes and gear.
- **Inputs and Outputs**
  Provide inputs to other agents and get the results from them which correspond to the intended action in the current frame.
- **Update the game state**
  After every frame the game situation is changed. The driver agent is responsible to update the game state in order to have access to updated parameters of the track, own car and opponents.

5.3.2 Track Agent

Track Agent determines the nature of the track by analyzing its various sections at run time. In order to have a sense of "How the track looks like in front and sides of the car", it divides the segments into straight, left curve, right curve and boundaries. In case of our solution, the track agent is responsible for keeping the car on track. Regardless of the track section, we are supposed to avoid the track boundaries or edges. For this purpose, we put repelling charges at the track edges in the field of track for boundaries (FOTB), which will help our car controller bot avoid the edges of the track. We need to put attractive charges on the track to drive along the track which is achieved through field of track (FOT). The formula to calculate these potentials are:

$$f_{FoTB}(c,d) = \frac{c}{\sqrt{d}} \dots\dots\dots\dots\dots\dots\dots\dots eq(4)$$

Where

- c = weight of the charge.
- d = distance between the lookahead position and position of the charge

$$f_{FoT}(m, w) = \frac{m}{w} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots eq(5)$$

Where

- m = distance to middle of the track.
- W = width of the track

The placement of repelling charges as well as the attractive charges is illustrated in the Figure 7. The track boundaries marked with negative signs are populated with repelling charges while the track middle with darker blue titles represent positive charges spread all along the track straight segment. The lighter sky blue charges represent charges with low potential on the sides of the track segment.

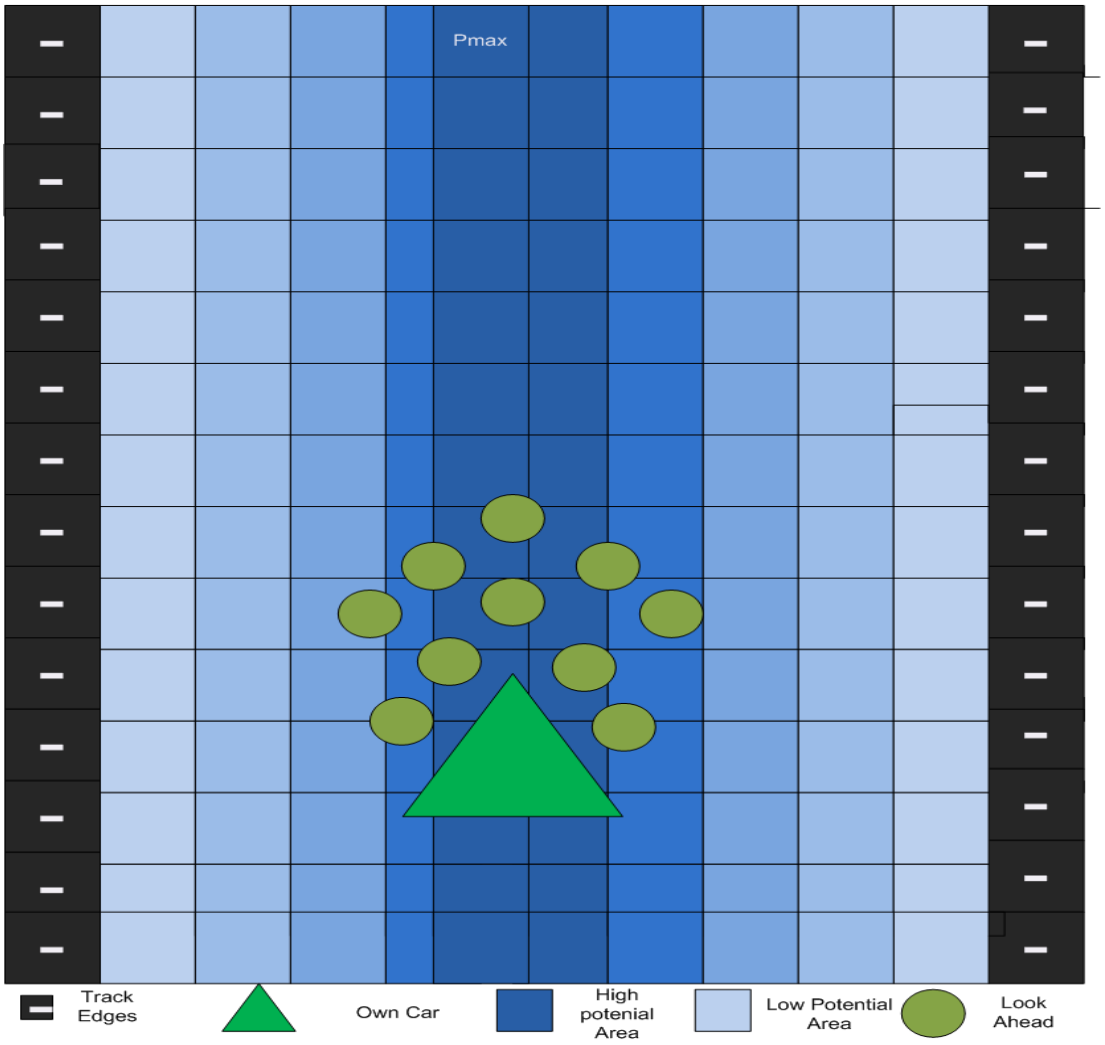


Figure 7: An illustration of Field of Track.

5.3.3 Curvature Agent

Curvature Agent plays a major rule in minimizing the laps time. This agent helps calculate the shortest path and minimize the curvature along the track curves. It is responsible for two kinds of fields in our solution i.e. Field of Curve for Right Turn (FOCR) and Field of Curve for Left Turn (FOCL). The two fields are similar in nature except for the exception in direction of the curve. In case of right turn the attractive charges are placed along the left side of the curve and vice versa. The track analysis described in section determine the type of track segments i.e. straight, left curve, right curve or boundary which is then used by curvature agent to put charges in appropriate positions. The formulas to calculate charges are

- Right Curve

$$f_c(m, d_{curve}, o, n) = m + d * o * e \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots eq(6)$$

Where

- f_c = charge in the field of curvature.
- m = vector to middle of the track.
- d_{curve} = direction of the curve
- o = possible correction to target point.
- e = vector to the end of the segment where target point is located.

- Left Curve

The formula is same for the left curve as described in the equation 3 except the value of d_{curve} which changes with direction of the curve. For the right curve segment, its value is -1.0f while for the right curve segment, it is 1.0f.

The potential for the charge in the field of curvature is calculated by the following formula

$$f_{FoC} = c\left(1 - \frac{d}{w} * o\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots eq(7)$$

Where

- w = width of the road
- d = the length between the positions of the LA point and the placement of the charge.
- o = correction to target point.

The Figure 8 shows the charges and associated fields where the green triangle represents own car, the C symbol placed on sides of the track which represent curvature charges on left and right sides. Each of these C exists at one time i.e. if the curve turns towards right the C is placed at left and vice versa. The mall circles around the triangle (own car) represent

lookahead points in front of the car. The darker blue titles represent high potential areas while the light blue titles represent low potential areas.

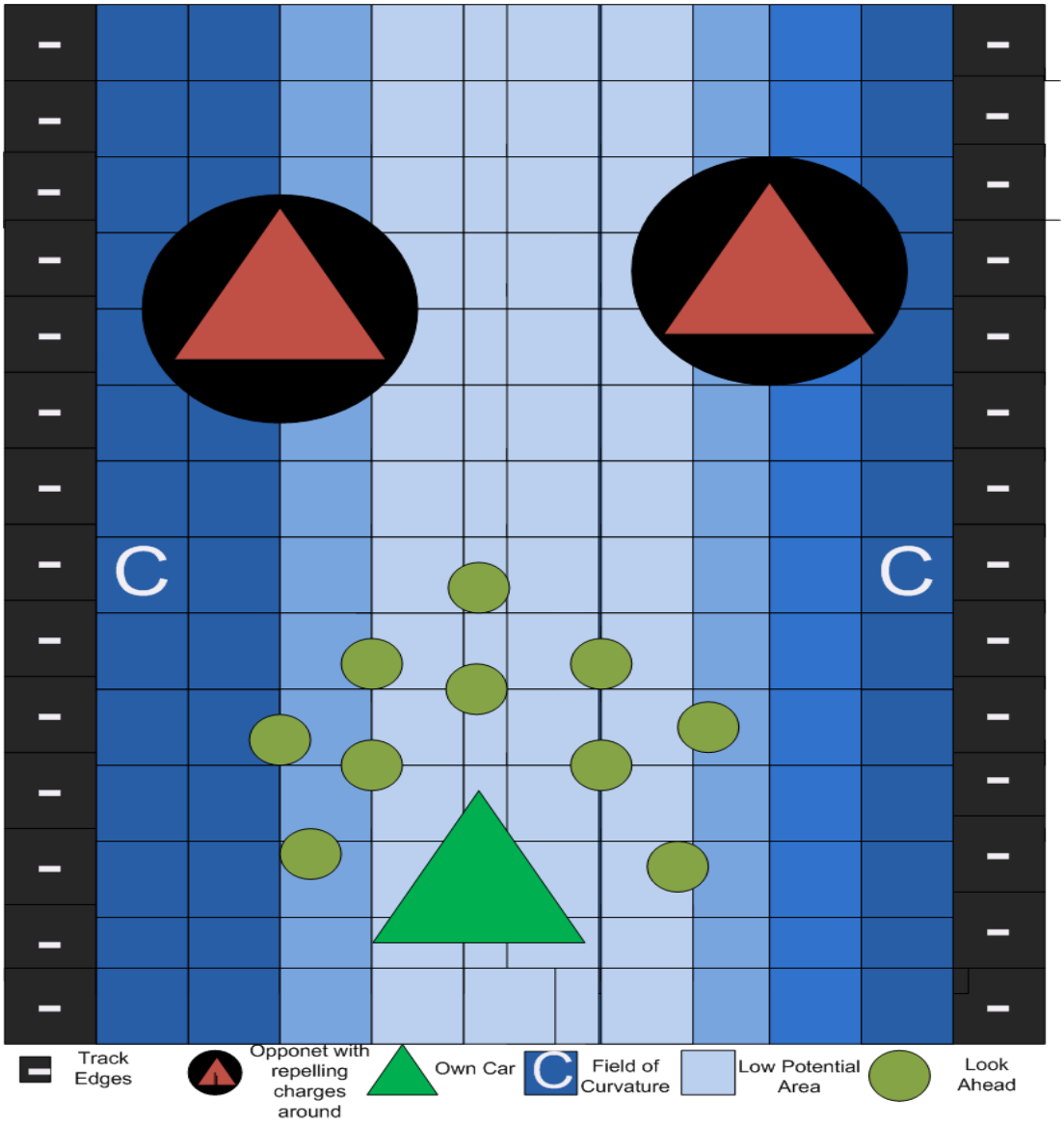


Figure 8: An Illustration of the Field of Curvature FOCL and FOCR for both the right and left curves of the track vice versa.

5.3.4 Speed Agent

Speed Agent is responsible for calculating allowed speed. As the track consists of different segments with varying speed requirements, it is important for our controller to maintain speed limits where there is required. Experienced drivers know the fact that there are limits for speed while driving along the curves or turns. These limits can be described and proved with laws of physics. In order to minimize the lap time we need to increase the speed in turns as much as possible without losing the control of the car. Centrifugal force plays a major role in pushing the car away from the center of turn towards outside of the turn due to high speed. To overcome the centrifugal forces and keep the car on track we used the friction on the

track and mass of the car into consideration. Traction also plays an important role in controlling such a phenomenon. The difference in traction of different wheels give rise to less road grip that disturb the normal steering control and compromise the stability of the vehicle. We solve the problem with the following equations.

$$a = \frac{v^2}{r} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots eq(8)$$

$$v = \sqrt{r * a} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots eq(9)$$

Where

r = radius of the track segment
a= acceleration
v= velocity of the vehicle

We also consider the fraction of the wheels on the track, gravitational force and mass of the car, the resulting equation therefore becomes.

$$v = \sqrt{a * r * m * g * \mu} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots eq(10)$$

Where

m = mass of the car
g = gravitational force
μ = fraction of the wheels on the track.

5.3.5 Overtaking Agent

Overtaking agent is responsible for the field of opponents (FOO). It was possible to assign different agent to opponents that handles all the aspects related to them but due to its possible interaction with our car in the race, we decided to handle opponents with overtaking maneuvers. In order to overtake, we need to keep notice of the following:

- Opponent to overtake
- Avoid collision with opponents
- Modify the steering value for overtaking
- Increase the speed
- Modify the target point

We first identify the side at which we want to overtake the opponent. When our car approaches near an opponent at a predefined minimum collision distance d_{col}, we put a repelling charge with negative potential field (FOO) at the intended side. The purpose of the FOO is to avoid collision with the opponent. The steer value is modified with modification of target point calculated by track and curvature agents. Target points represent attractive charges with positive potentials to guide the controller drive along the track in intended direction. Due to overtaking maneuvers and collision avoidance an offset is added to target points to relay the change in direction perpendicular to tangent angle of the track. The formula to calculate the FOO and speed are:

$$f_{Foo}(s, c, d_{col}) = c(1 - s * d_{col}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots eq(11)$$

Where

- c = is the repelling charge value
- s = is the side of the opponent
- d_{col} = is the minimum collision distance.

5.3.6 Traction Control Agent

Traction Control Agent is responsible for traction control system (TCS). Traction control system (TCS) is used to avoid loss of traction due to driven road wheels. It is also known as anti-slip regulation (ASR) and widely used as a secondary function to anti-lock braking system (ABS) on vehicles. If properly implemented, it enhances driver control when applied throttle is miss-matched with road surface conditions due to brake force on one or more wheels. A traction control system is put in place when steering control and stability of the vehicle is compromised due to loss of road grip. Driving along the curves at high speed, the outer and inner wheels of the car have different rotation speed that creates a difference in slip. In order to overcome the effect of slip and difference between the rotation speed of the wheels a technique called differential is used [17]. The formula to calculate differential in our case is:

$$f_{TCL}(accel, cs) = \{accel = accel - MIN\ (accel, (slip\ -\ \alpha)/\beta)\ if\ slip > \alpha\ \} \ldots\ldots\ldots eq(12)$$

And

$$slip = w_{ds} - v_c \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots eq(13)$$

Where

- f_{TCL} = Traction control function.
- accel = Current acceleration.
- cs= Car State
- α = Traction control slip constant.
- β= Traction control range constant.
- w_{ds} = Driven wheel speed from the WheelSpinVal sensors.
- v_c = Current velocity

5.3.7 The Interface Agent

The interface Agent as its name indicates connects the agents with the TORCS server. It provides sensory information to the agents of the system about the track, own car and opponents. It manages the output of the agents so that TORCS is able to understand what action the agent wants to take. It manages the input and output in connection with TORCS and other agents in the system.

5.3.8 Track Analysis Agent

Simulated Car Racing Championship Software only provides sensory information, therefore it is necessary to analyze the track and identify the portions of track that will be later used in the placement of charges. TORCS server provides this information via tTrack structure in the track header i.e. the track is divided into segments and each segment is either straight, left curve or right curve. While programming bot for the TORC server it is relatively easy to place charges on the segments due to availability of complete track information but in simulated car racing championship software we analyze the track ourselves to get that necessary information. We are provided warm up races in the competition to drive along the different tracks and analyze the track for further rounds. We drive along the tracks at very low speed and analyze the track while recording points of interest on the track. The recorded points of interest help us find out whether a curve is coming next and if yes what's the direction of the curve. It helps us to place the charges at appropriate positions on the track while we can minimize curvature and maximize speed as well. The strategies to determine left turn, right turn, and straights are:

- Left and Right Turns:

  The front sensors that consist of front right and front left sensors pointing at -5 and 5 degrees respectively help us identify the direction of the curve. The k-value of the line between these two sensors determines the direction of the turn. If k-value is negative, the turn is considered left otherwise in case of positive values, it is considered a right turns.

- Straights:

  The lengths determined by the sensors at the sides of our car (right, left) in comparison with the length measured by the front sensors help us identify the straights. If the length returned by the frontal sensors is greater than the length returned by the sensors at sides, we consider the segment as straight.

5.3.9 Determine left/right placement of latitude position

The width of the track in TORCs remains constant throughout the length of the track which makes it easy to use it for every segment of the track. In Simulated Car Racing Championship Software, variations may occur due to noisy sensor's data which is used to determine the width. The width of the track segment is determined by adding the values recorded from sensors spanning from -90 to 90 degrees. The width is then used along with track position provided by the competition software to measure the distance to track edges. The formula to calculate the right and left edges are:

$$d(trkPos, w) = \begin{cases} (0.5 + 0.5) * w & if\ trkPos\ > 0 \\ (0.5\ + 0.5 * (trkPos\ - 1.0) * w) & if\ trkPos\ < 0 \end{cases} \dots eq(14)$$

Where

- trkPos = current track position.
- w = width of the road.

The above mentioned formula calculates distance to the right edge of the track segment. Now when we have this, we can easily calculate the left edge by subtracting it from the width of the track. The formula is:

$$d(w, d_{right}) = w\ -\ d_{right} \dots\dots\dots\dots\dots\dots\dots\dots\dots eq(15)$$

Where

- d_{right} = distance to right edge to the track.
- W = the width of the track.

6 EXPERIMENTS

The experiments are conducted both on TORCs server itself and Simulated Car Racing Championship Software. In the case of Simulated Car Racing Championship Software, the TORCs server waits for the client to connect and begin the race while in TORCs sever (standalone environment), the dynamic link libraries are provided to enlist our controller. We used the same computer for the server and client version of the experiment. The specification of the computer used is depicted in the Table

Items	Description
Windows edition	Windows 8 Pro
TORCs Version	Version (1.3.4)
Simulated Car Racing Championship	
Processor	AMD Athlon™ 64X2 Dual-Core 1.73 GHz
Installed memory (RAM)	2.00 GB
System Type	64-bit Operating System, x64-based processor

Table 2: Specification of the computer used in the experiment

The experiments demonstrate the working of different agents, their interaction with one another and the positive impact of its modular approach on overall performance of the controller. We evaluate the performance of our controller in terms of lap times, number of damages, position in the race and effective use of artificial potential fields. The performance measures obtained are then compared with that's of opponents to get the ultimate results.

6.1 Brief description of the opponents

For the experiments we have used the last year competitors Autopia which was the winner from 2010 simulated car racing championship, COBOSTAR which was the winner from 2009 simulated car racing championship, and Neil which finished around the middle placement in the simulated car racing championship 2010.

6.1.1 Autopia

Autopia is one among the opponents for our controller in the competition which uses somewhat fuzzy architecture. It is without no doubt a competitive controller with fast speed and accurate sense of tracks. It consists of three basic modules comprising gear change, target speed along the different segments and steer control. It keeps records of track positions throughout the length of the track in an initialized vector that changes in certain defined situations e.g. when the car is out of track, the vector positions are multiplied with 0.95 before reaching the current position which is considered 250 meters ahead. It counts the damages during the current lap of the race and adopts a cautious strategy for the next lap. The function that maintains the vector is:

$$f(v) = 1 - 0.02 \ \ if \ n_d = 1000$$

Where

- n_d =number of damages during the warm-up.

6.1.2 Olethros

The car controller with the name Olethros is one among the fastest driver provided by the TORCS engine. It uses an estimation system for controls based on predictions. It is based on bt controller developed by Bernard [16] but differs on some estimated quantities. It uses four types of different estimates which are:

- Estimate of future steering/acceleration.

The controller uses the default value for steering and acceleration provided by the skeleton of the controller it relies on in startup but updates these values at every timestamp using the following formulas.

$$aceel(t-k) \mathrel{+}= pow(\lambda, k) * a * \big(accel\_input(t+1)\big) - accel(t)$$

And

$$steer(t-k) \mathrel{+}= pow(\lambda, k) * a \\ * \big(steer\_input(t+1) + \gamma * steer(t+1) - steer(t)\big)$$

Where

- λ = the values of lambda is in the range [0, 1]. The smoothing effect depends upon the value of lambda, as its value approaches near to 1; the smoother effects become visible on acceleration.
- γ = the values of Gamma is in the range [0, 1]. The steer commands are modified when likewise with Gamma values when it approaches near to 1.

- Estimate of Curve Radius.
- Estimate of friction of brakes.
- Danger estimates.

6.1.3 Inferno, Lilliaw and dammned

These controllers are developed by Bernard [16] and integrated with the TORCS server from time to time. These bots are earlier versions of Bernard's bt controller bot with different strategies and slighter modifications. These are comparably slow but were later modified into bt controller bot with advance machine learning and car control solutions like aerodynamics and traction control measures.

6.2 The Tracks

TORCS provide a number of tracks from complex to simple one. The complexity is provided in the form of tricky turns, fast corners, high speed track segments and number of other important parameters like width and length etc. we have used both simple and complex tracks to evaluate our controller. The tracks along with their description are depicted in Table 3. Figure 9 and 10 shows how the different track looks like in drawings.

Track	Description	Author	Length	Width	No of Pits
CG Speedway number 1	Quit Fast paced track	CG, B. Wymann	2057.56 m	15.00 m	20
CG track 2	High Speed track	CG	3185.83 m	15.00 m	16
Ruudskogen	Simple track	A.Sumner	3274.20 m	11.00 m	16
Wheel 2	Fast curvy track	A.Sumner	6205.46 m	12.00 m	28
Aalborg	Medium complexity track	E.Espie,B.Wymann	2587.54 m	10.00 m	16
Alpine 1	Alpine track	E.Espie	6355.65 m	12.00 m	16
Alpine 2	Slow mountain road	D.Schellhammer	3773.57 m	10.00 m	16
Brondehach	Technically demanding circuit	A.Summer, Milestone	3919.31 m	13.00 m	16

Table 3: Tracks along with description

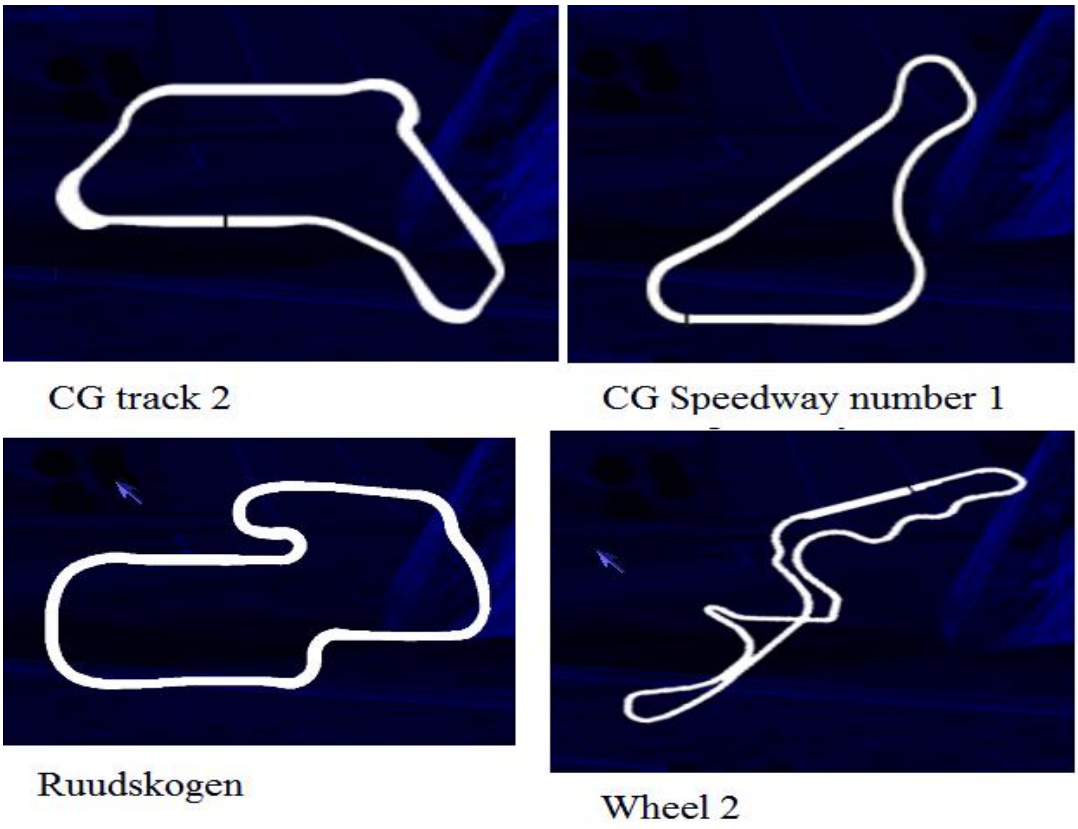


Figure 9: Maps of CG track 2, CS Speedway number 1, Ruudskogen and Wheel 2

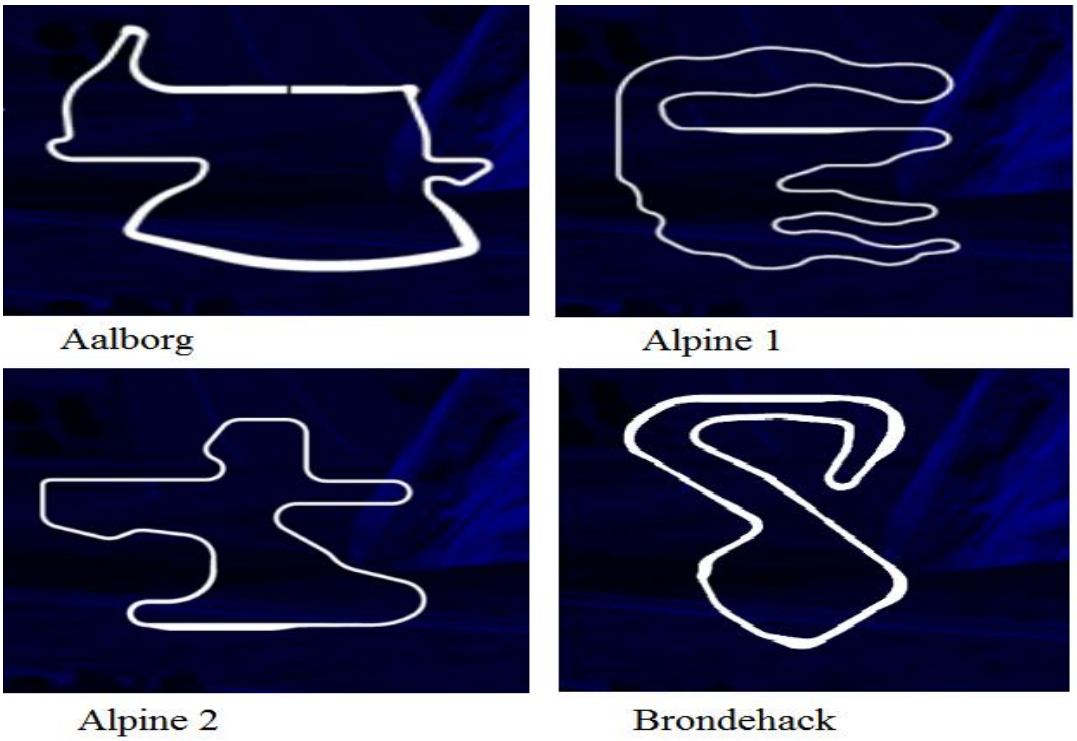


Figure 10: maps of Aalborg, Alpine 1, Alpine 2 and Brondehack tracks

6.3 Preparation

Each controller bot is provided an opportunity in the competition to drive along the selected track in warm up race to learn the track and if required record the points of interest. All controllers are given 100,000 in game ticks for warm up race. We used 6 different tracks from simple to complex one for the practice or warm up session. Our controller drives along the tracks in the practice session in order to learn the track and record laps time at different tracks. We conducted three experiments which are described in the sections 6.4, 6.5 and 6.6 respectively.

6.4 Experiment 1

In this experiment, we illustrate the effects of agents on the overall outcome of the race using lap times, speed and number of damages during the race. The controller drives along each track without opponents starting from the pole positions of the tracks. There is an option provided in the TORCS server where the number of laps could be set for the controller. We selected 10 laps for this experiment in order to record as much data as required for analysis. it is important to mention here that the controller developed in [1] was unable to drive without activating the shortest path agent therefore each active agent was working alongside shortest path agent in their experiment for the same purpose. In case of our controller all the agents work independently as well as in collaboration with the other agents of the system.

6.4.1 Track Agent

First we activate the track agent in our experiment and disable all other agents in order to record its performance parameters in terms of total lap time, best lap time, top speed,

minimum speed and number of damages during the practice session. On some tracks such as Wheel 2, Alpine 1 and Alpine 2 the lap time is invalidated that's why they are marked N/A in the Table 4. The lap time is invalidated because track agent requires the help of other agents in the system to drive along these complex tracks. The performance parameters and their results for our controller are depicted in Table 4.

Track	Total Time	Best Time	Top Speed	Min Speed	Damages
CG Speedway number 1	453:62	44:58	273	114	0
CG track 2	N/A	N/A	N/A	N/A	N/A
Ruudskogen	760:43	73:43	243	62	0
Aalborg	881:36	87.44	224	45	0
Wheel 2	N/A	N/A	N/A	N/A	N/A
Alpine 1	N/A	N/A	N/A	N/A	N/A
Alpine 2	N/A	N/A	N/A	N/A	N/A
Brondehack	982:21	95:52	247	56	62

Table 4: The results of having only track agent activated.

6.4.2 Curvature Agent

In this section of the experiment only curvature agent is activated. The curvature agent in our system is responsible for both the shortest path and minimal curvature unlike the controller developed in [1] where shortest path and curvature agents are separate entities. The performance parameters recorded when only curvature agent was active are depicted in Table 5.

Track	**Total Time**	**Best Time**	**Top Speed**	**Min Speed**	**Damages**
CG Speedway number 1	424:06	41:08	244	113	0
CG track 2	575:01	54:86	276	104	32
Ruudskogen	632:22	67:89	246	77	6
Aalborg	N/A	N/A	N/A	N/A	N/A
Wheel 2	1197:85	116:39	283	34	990
Alpine 1	1482:80	142:58	257	35	0
Alpine 2	N/A	N/A	N/A	N/A	N/A
Brondehack	883:21	85:33	249	78	76

Table 5: The results of having only Curvature Agent activated.

6.4.3 Track Agent with Curvature Agent

In the previous sections Track Agent and Curvature Agent were active only one at a time. Now we activated both agents at the same time in order to find out their combined effect in the outcome of the race and on the performance parameters such as total time, best time and top speed etc. the results are depicted in Table 6. The data for certain tracks are marked with N/A , the lap's time on those track was invalidated because our controller did not manage to successfully drive along those tracks while track agent and curvature agent both are activated at the same time.

Track	Total Time	Best Time	Top Speed	Min Speed	Damages
CG Speedway number 1	424:06	41:08	244	113	0
CG track 2	575:01	54:86	276	104	32
Ruudskogen	632:22	67:89	246	77	6
Aalborg	N/A	N/A	N/A	N/A	N/A
Wheel 2	1197:85	116:39	283	34	990
Alpine 1	1482:80	142:58	257	35	0
Alpine 2	N/A	N/A	N/A	N/A	N/A
Brondehack	883:21	85:33	249	78	76

Table 6: The results of Track Agent in combination with Curvature Agent

6.4.4 All Agents

It is important to activate all the agents all together in order to validate how each agent carries out its work load and how it interacts with other agents of the system. In the previous sections we recorded data when individual agents and pair of agents were active at one time. We recorded the data when all the agents of the system are active at the same time in table 7. The number of laps was kept at 10 in order to record as much data as necessary. The lap's time on certain track is invalidated that's why they are marked with (N/A). Our controller does drive on those tracks but with lap's time invalidation.

Track	**Total Time**	**Best Time**	**Top Speed**	**Min Speed**	**Damages**
CG Speedway number 1	424:06	41:08	244	113	0
CG track 2	575:01	54:86	276	104	32
Ruudskogen	632:22	67:89	246	77	6
Aalborg	N/A	N/A	N/A	N/A	N/A
Wheel 2	1197:85	116:39	283	34	990
Alpine 1	1482:80	142:58	257	35	0
Alpine 2	N/A	N/A	N/A	N/A	N/A
Brondehack	883:21	85:33	249	78	76

Table 7: The results of all agents activated at the same time

6.4.5 Results

In the previous sections we recorded data when different agents and pair of agents were active at one time along with all the agents active at the same time. In this section we evaluate the results obtained in sections from 6.4.1 to 6.4.4 in the form of charts to show how the best lap's time, total laps time and top speed differ with activation of different agents among the system and the whole agent system of the controller in general. The Figure 11 shows Best Lap's time comparison along the different tracks with effects of different field of forces marked on the horizontal axis. The best lap's time meter is shown on the vertical axis of the graph. The name of the tracks is marked on the horizontal axis of the chart above the field of forces. The Best lap's time improves when all the agents are activated at once. In Figure 12 we make Total time comparison of data taken in the previous sections along the different tracks. The tracks are marked on the horizontal axis of the track along with field of forces below which represent different agents of the system. The total lap's time meter is placed at y-axis. In Figure 13 we evaluate different agents and pair of agents on the basis of top speed at different tracks. The tracks are placed at horizontal axis while the top speed meter is placed on vertical axis.

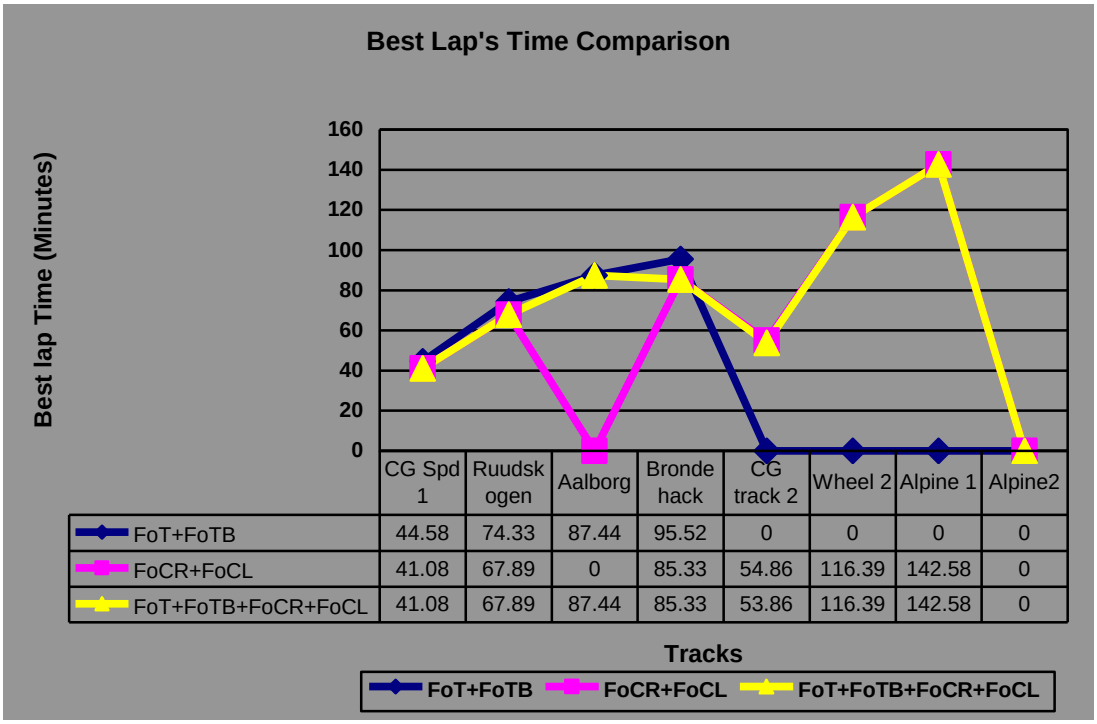


Figure 11: Best Lap's Time Comparison on different tracks among the different agents

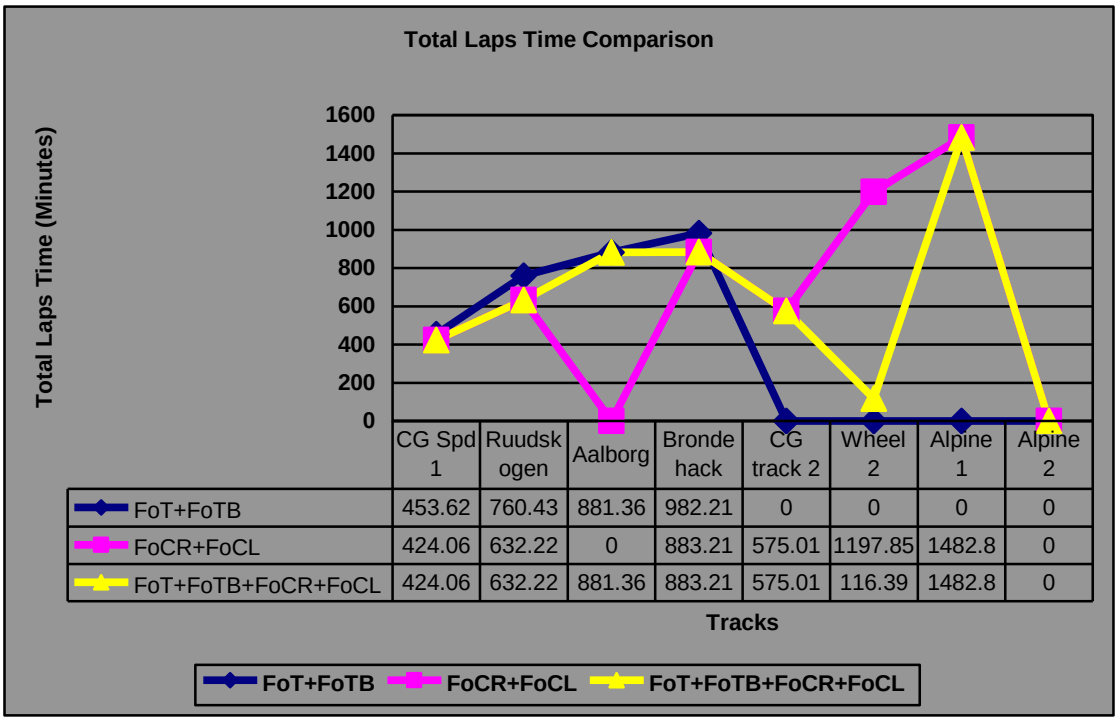


Figure 12: Total Laps Time Comparison on different tracks among the different agents

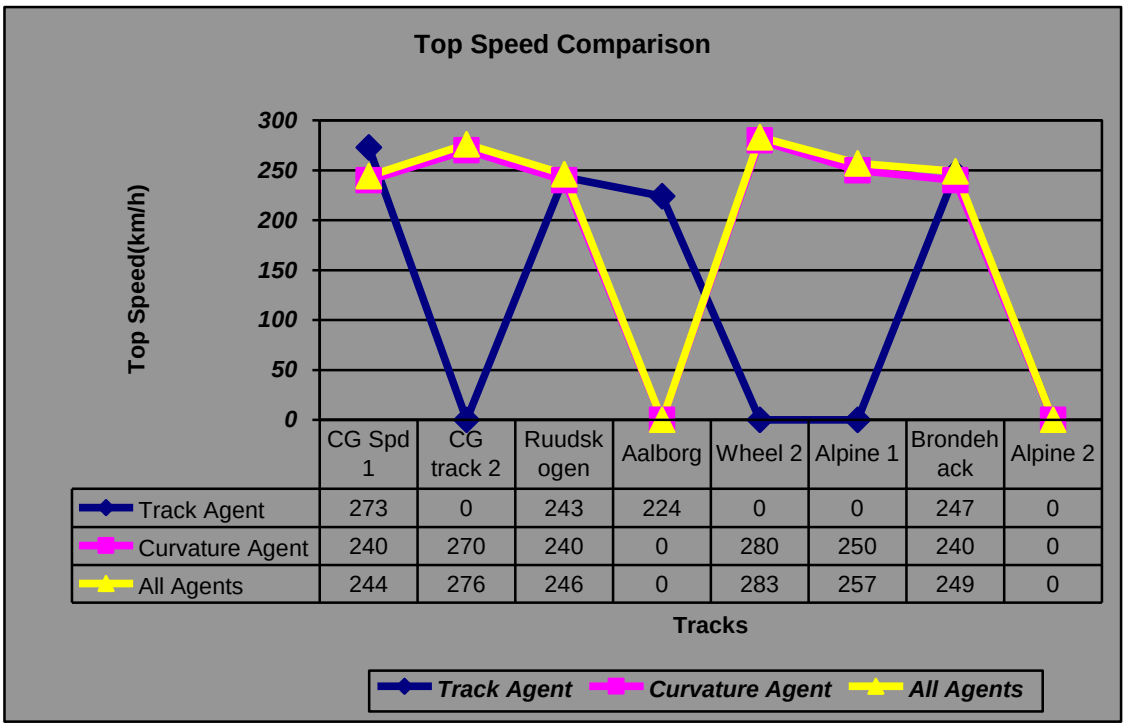


Figure 13: Top Speed Comparison on different tracks among the different agents

6.4.5.1 Comparison with Powaah

The driver controller developed in [1] is named "Powaah". We extend their work in our thesis but we named our controller "Eagle". Here we make the comparison between the old controller "Powaah" and the extended one "Eagle" on the basis of best lap's time on different tracks. The purpose of the comparison is to analyze the improvement made after introducing effective collision and overtaking strategies. The best lap's time of our controller "Eagle" on described tracks as eminent from the Figure 14, is considerably improved from "Powaah".

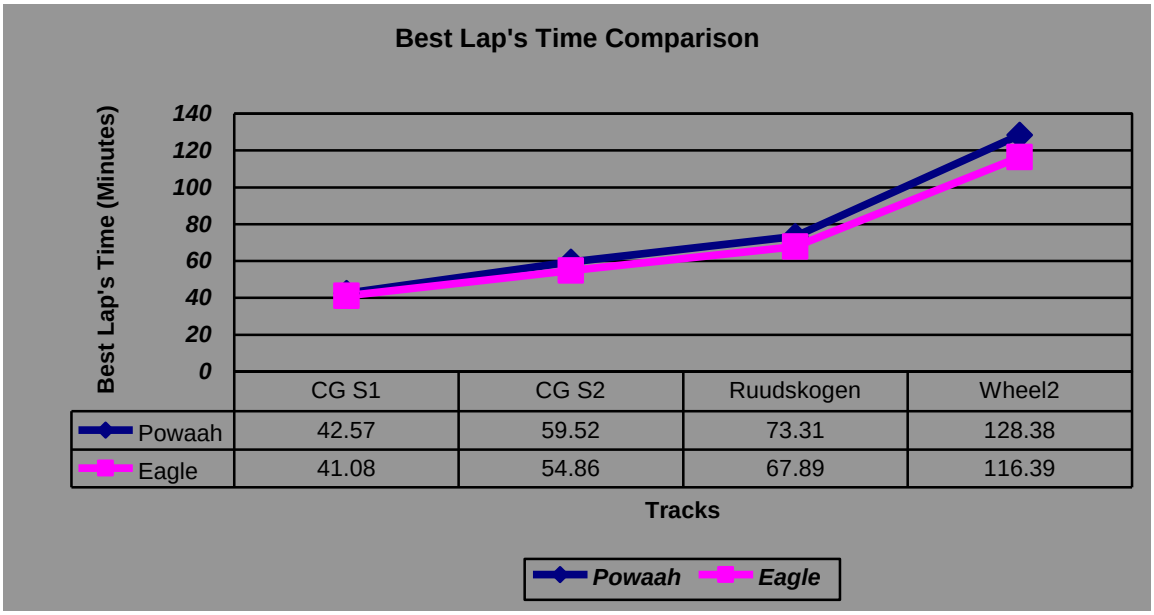


Figure 14: Best Lap's Time Comparison between our controller (Eagle) and the one we extended (Powaah) on different tracks

6.5 Experiment 2

In this Experiment we demonstrate the comparison between our controller and the opponents in terms of performance parameters such as position in the race, Best lap Time, Top Speed and number of damages etc. we used different tracks for the race and the results are compiled track wise. Every controller has some limitations regarding the tracks; some performs better on certain tracks while failing to complete the race on other tracks. The results of the race and their comparison along with graphical representation are presented in section 6.5.1, 6.5.2, 6.5.3, 6.5.4 and section 6.5.5 respectively.

6.5.1 CG Speed Way 1

Rank	Driver	Best lap Time	Total Time	Top Speed	Damages
1	Olethros	41:40	423:36	252	1086
2	Eagle	40:82	480:31	246	2006
3	Inferno	41:08	720:45	249	340
4	Lilliaw	41:24	900:35	249	963
5	Autopia	42:84	1860:23	242	1396
6	Damned 6	47:39	+1 Lap short	208	0

Table 8: Results of the race on CG Speed Way 1

6.5.2 CG track 2

Rank	Driver	Best lap Time	Total Time	Top Speed	Damages
1	Eagle	55:64	580:53	274	527
2	Oltheros	58:76	1380:60	274	1654
3	Lilliaw	54:27	1440:98	277	2730
4	Inferno	56:80	+1 Lap short	268	1722
5	Damned	65:47	+1 Lap short	224	807
6	Autopia	164:58	+6 Lap short	249	2522

Table 9: Results of the race on CG track 2

6.5.3 Wheel2

Rank	Driver	Best lap Time	Total Time	Top Speed	Damages
1	Oltheros	104:93	580:53	285	10
2	Damned	104:56	1380:60	290	77
3	Inferno	105:61	1440:98	297	727
4	Lliaw	105:57	+1 Lap short	287	1099
5	Eagle	105:66	+1 Lap short	289	2079
6	Autopia	00:00	+10 Lap short	109	9208

Table 10: Results of the race on Wheel 2

6.5.4 Alpine 1

Rank	Driver	Best lap Time	Total Time	Top Speed	Damages
1	Oltheros	133:64	2688:15	270	878
2	Eagle	142:41	106:78	256	428
3	Damned	144:04	140:21	250	1764
4	Lliaw	133:60	140:39	286	516
5	Inferno	144:16	141:42	266	4234
6	Autopia	00:00	+10 Lap short	182	10526

Table 11: Results of the race on Alpine 1

6.5.5 Quick Race Results with Opponents

In the previous sections we recorded data when a number of car controllers including our controller "Eagle" race with each other in quick race. In this section we evaluate the results obtained in sections from 6.5.1 to 6.5.4 in the form of charts to show the best lap's time, total laps time and top speed along with position in the race of the competing car controllers. The Figure 15 shows Best Lap's time comparison between the competing car controllers along the different tracks. The best lap's time meter is shown on the vertical axis of the graph while the controller names are displayed on vertical to the right. We achieved best lap's time on CG track 2 and got 1st position in the race. We are 2nd in best lap's time as well as in position in the race on CG Speed way 1 and Alpine 1. Our controller shows the worse performance on wheel 2 which is technically tricky and complex track. In Figure 16 we evaluate different controllers along with our own car controller on the basis of top speed at different tracks. The tracks are placed at horizontal axis while the top speed meter is placed on vertical axis. The car controllers with their names are displayed on the vertical to the right side of the chart.

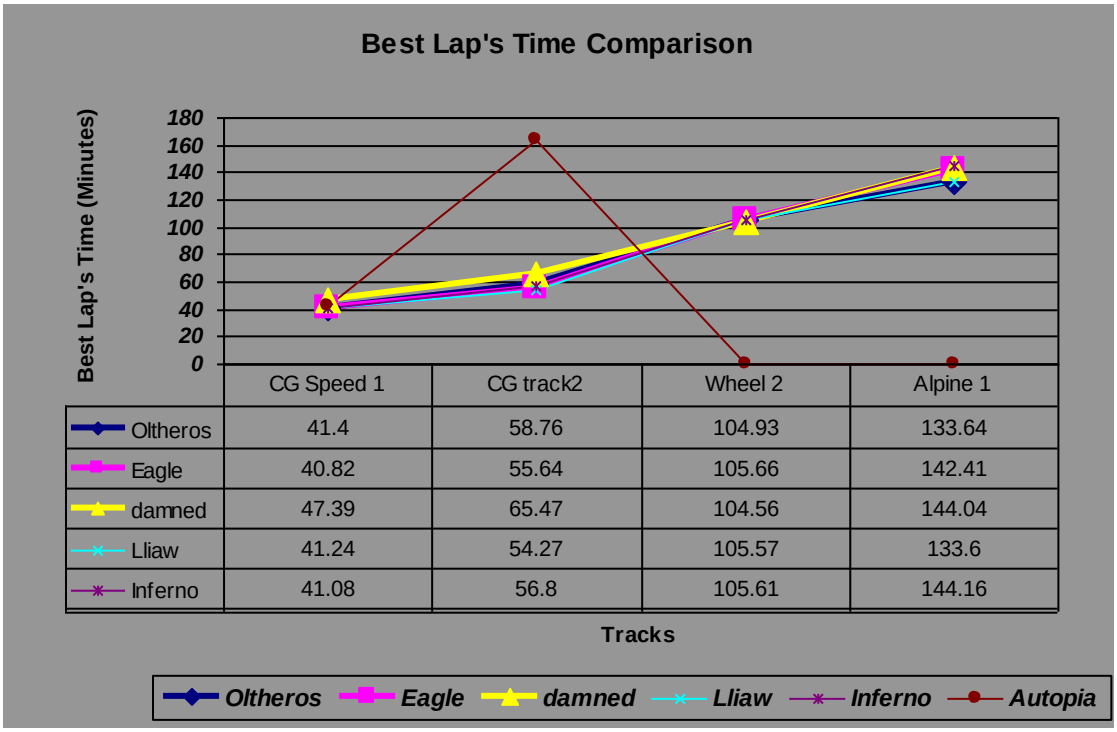


Figure 15: Best Lap's Time Comparison between our controller (Eagle) and other opponents

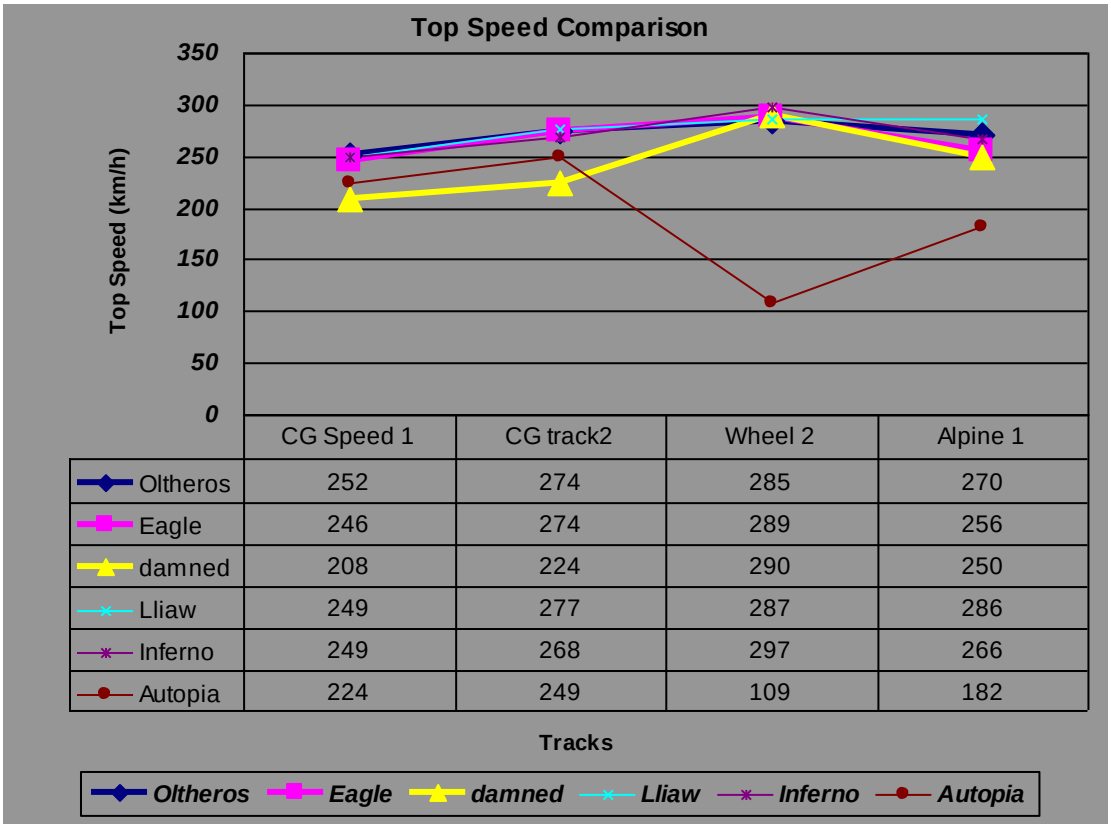


Figure 16: Top Speed Comparison between our controller (Eagle) and other opponents

6.6 Experiment 3

In this experiment we disabled Field of Forces for Opponents (FOO) in order to record how much damages we get along different tracks while competing with opponents. As described earlier that Field of Forces for Opponents are used for overtaking and collision detection with the opponents, driving without it will provide us damage data to compare it with the similar data taken in experiment 2 when Field of Forces were enabled for our controller "Eagle".

6.6.1 CG Speed Way 1

Rank	Driver	Best lap Time	Top Speed	Damages
1	Oltheros	41:06	254	299
2	Eagle	44:68	297	1274
3	Damned	46:29	212	0
4	Lliaw	33:04	297	0
5	Inferno	32:88	298	0
6	Autopia	42:84	242	1396

Table 12: Results of the Quick race with opponents without field of opponents (FOO)

6.6.2 CG Track 2

Rank	Driver	Best lap Time	Top Speed	Damages
1	Oltheros	58:37	280	990
2	Eagle	62:99	264	5264
3	Damned	67:43	225	4025
4	Lliaw	42:28	325	1453
5	Inferno	42:12	325	1339
6	Autopia	164:58	249	2522

Table 13: Results of the Quick race with opponents without field of opponents (FOO) on CG Track 2

6.6.3 Wheel 2

Rank	Driver	Best lap Time	Top Speed	Damages
1	Oltheros	114:56	286	139
2	Eagle	138:43	271	11678
3	Damned	135:24	236	350
4	Lliaw	94:05	327	84
5	Inferno	95:37	325	195
6	Autopia	00:00	109	9208

Table 14: Results of the Quick race with opponents without field of opponents (FOO) on Wheel 2

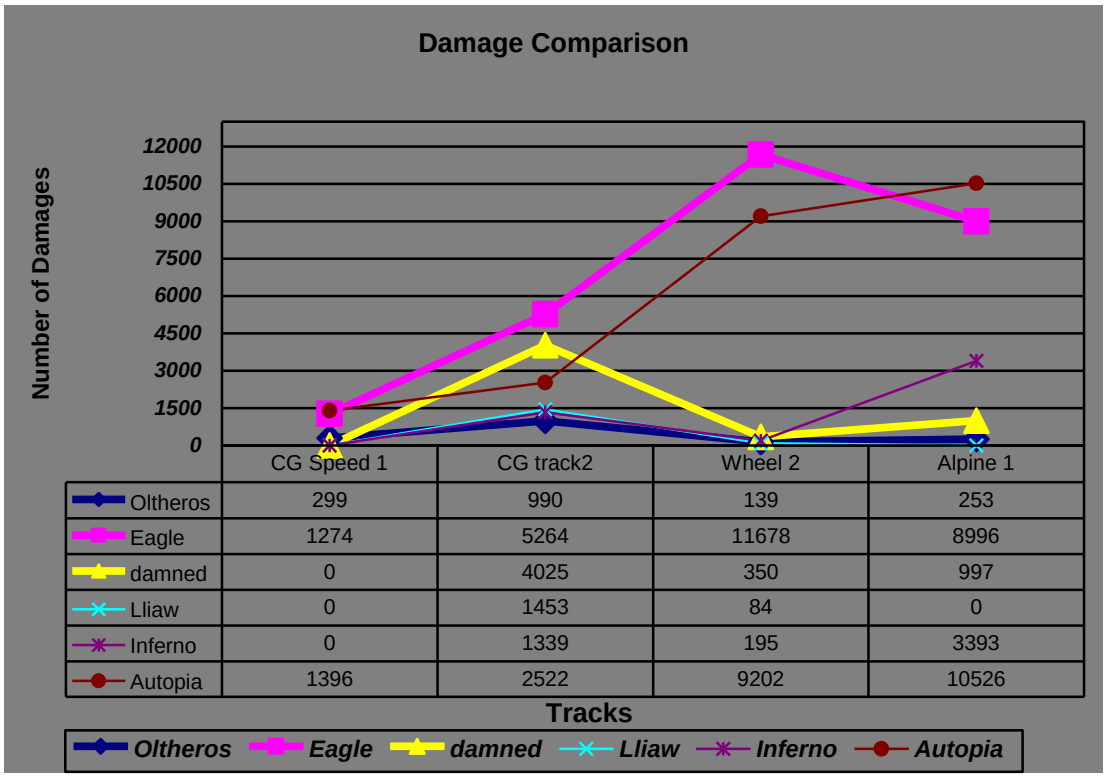
6.6.4 Alpine 1

Rank	Driver	Best lap Time	Top Speed	Damages
1	Oltheros	134:22	270	253
2	Eagle	164:33	235	8996
3	Damned	162:52	209	997
4	Lliaw	112:91	328	0
5	Inferno	112:52	307	3393
6	Autopia	00:00	182	10526

Table 15: Results of the Quick race with opponents without field of opponents (FOO) on Alpine 1

6.6.5 Damage Comparison (FOO Disabled)

When we study the graph in Figure 17, it is clear that our car controller got higher number of damages than the other competing car controllers. The reason is absence of Field of Forces for Opponents (FOO) which keeps our controller at a safe distance from opponent by putting repelling charges at the sides of the car.

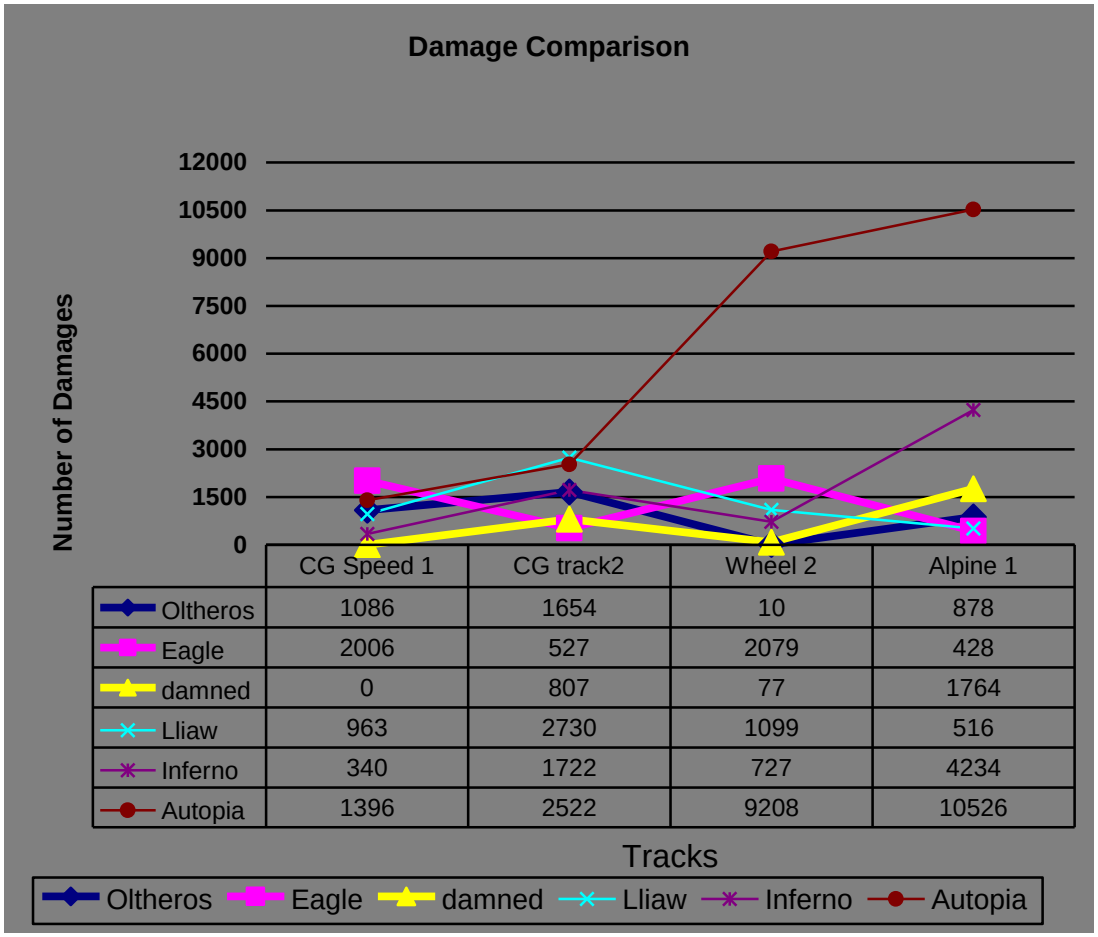


	CG Speed 1	CG track2	Wheel 2	Alpine 1
Oltheros	299	990	139	253
Eagle	1274	5264	11678	8996
damned	0	4025	350	997
Lliaw	0	1453	84	0
Inferno	0	1339	195	3393
Autopia	1396	2522	9202	10526

Figure 17: Damage Comparison between Eagle (FOO disabled) with that of opponents

6.6.6 Damage Comparison (FOO Enabled)

The damage data taken in experiment 2 with Field of Forces (FOO) enabled is here by compared with that of opponents in Figure 18. Here we recorded less number of damages as compare to section 6.6.6 (Figure 17) when Field of Forces for Opponents (FOO) were disabled. Overtaking is a complex phenomenon which might lead to increase number of collisions if not properly handled thus resulting in higher number of damages and lower position in the race. The data in the Figure 18 reveals that our controller was able to reduce the number of damages taken during overtaking the opponents in the race using artificial potential fields.



	CG Speed 1	CG track2	Wheel 2	Alpine 1
Oltheros	1086	1654	10	878
Eagle	2006	527	2079	428
damned	0	807	77	1764
Lliaw	963	2730	1099	516
Inferno	340	1722	727	4234
Autopia	1396	2522	9208	10526

Figure 18: Damage Comparison between Eagle (FOO Enabled) with that of opponents

7 DISCUSSION

The idea of Artificial potential fields originates to solve real world problem when Khatib [3] used the concept for collision detection in manipulators and robots. The previous work done by Rossetter [14] which successfully demonstrated the use of artificial potential fields for accident avoidance from lane departure in real cars. The concept is also tested in virtual game environment [7] [1]. simulated car racing championship is being used as a test bed for the concept of multi agent based artificial potential fields in car racing games [1] but did not manage to demonstrate the use of artificial potential fields for collision avoidance and overtaking. In our thesis, we used the concept of MAPFs (Multi-agent Artificial Potential Fields) for collision avoidance with the opponent's car as well as with the track edges and for overtaking the opponents during the race. We conducted a number of experiments and the results show the multi agent based artificial potential field approach bitterly suit the car racing game environment. Collision detection is always considered a tricky and technically challenging task in rapidly changing environments like the simulated car racing championship [6]. To achieve the functionality of collision detection among dynamic objects using artificial potential field makes the task more challenging because of the incomplete information due to rapidly changing environment. Despite of its complexity, it delivered wonderful results and successfully avoid collision with the track edges as well as with the competing opponents.

Our test results in experiment 1 and 2 reveals that APFs very well fit the domain of car racing game. In multi agent based system although the agents are considered autonomous and act on their own with or without the influence of other agents in the system but in order to achieve better performance all the agents must work in collaboration with each other while preserving the unified structure of the system. In Experiment 1, we activate a number of agents and pair of agents while recording the performance data likewise. The results shows that the performance is better when all the agents are activated in once as a whole. When we had all the agents activated, the car follow the shortest path possible while minimizing the curvature to maximize the speed in curves thus resulting in fewer lap's time, faster speed during turns and influence the race results.

In Experiment 2 our bot compete with other participants controllers of the race. Our controller yield better performance on CG track 2 by defeating the oltheros, which is fast and efficient controller and considered one among the best controllers provided by the TORCS server. On CS Speed way 1 and Alpine 2 we stood at 2nd position after oltheros but failed to get position on wheel2. The problem with the car racing simulator like TORCS is the lack of generalization for the controller. One needs to optimize the performance on every track in order to show better results as compare to the opponents.

Thus with our experiments we have shown that multi agent based artificial potential fields approach not only offer a modular way to solve the complex and tricky problems of game racing but also fit to compete with the traditional methods used so far for collision avoidance, path finding and overtaking.

8 CONCLUSION AND FUTURE WORK

8.1 Conclusion

It is important to mention once more time the hypothesis we intended to prove in order to conclude the thesis work,
"**Does a Multi-agent based APFs (Artificial Potential Fields) solution provide better results than the traditionally used methods for collision detection and overtaking in a car racing games**?"
We have concluded by implementing a controller that make use of multi agent based artificial potential field approach to achieve the tricky and complex task of collision detection and overtaking while driving in a car racing game with different other controllers as opponents. We exploited the advantages of APFs to the best of our knowledge using laws of physics and discrete mathematics in order to achieve successful overtaking behavior and overcome its drawbacks as being very complex to implement and high memory requirements for time critical applications e.g. car racing games (Section 3.1, RQ1). Dynamic objects in a fast changing environment like a car racing game are likely to collide more often with each other, thus resulting in higher number of damages. Using APFs instead of traditionally used collision avoidance techniques resulted in less number of damages during the race, thus minimizing the lap's time which in turn contribute to better position in the race as shown in experiment 3 (Section 3.1, RQ2 and Section 6.6). Overtaking maneuvers are complex and tricky and is the major cause of collision among cars both in real life as well as in car racing games, thus the criteria to measure the performance regarding overtaking behavior of different controllers in the race is based on number of damages taken during the race. The comparison between the participating controllers in terms of damages taken during various rounds of the race is analyzed in experiment 3 (Section 3.1, RQ3 and Section 6.6). The results of the quick race along with opponents shows good results on three tracks while having bad performance on the remaining other track. Our controller got 1st position on the CG track 2 while kept 2nd position on CS Speed way 1 and Alpine 1. It had worse performance on wheel 2 which needs to be optimized in the future for better results on this track and other similar complex and tricky tracks.

8.2 Future Work

In simulated car championship software [6], the track information provided is illusive and incomplete. We had to use sensors even to determine the length and width of the different segments. This led us to use artificial potential fields for specific positions on the track instead of populating the whole track with attractive and repulsive poles like the work of Hagelbäck and Stefan J. Johansson [7]. The TORCS [2] server itself provide complete information about the track ranging from track boundaries (edges) to different segments of the track weather straight, left or right segments. The track class that incorporates the track information also provides segment's vertex information which could be used to populate the whole track with field of forces thus paving the way for moving the car in field of forces along the track. The Figure 17 shows some general Track information provided by the track header.

```
    tdble startWidth;       /**< Width of the beginning of the segment */
    tdble endWidth;     /**< Width of the end of the segment */
    tdble lgfromstart;      /**< Length of begining of segment from starting line */
    tdble radius;       /**< Radius in meters of the middle of the track (>0) */
    tdble radiusr;      /**< Radius in meters of the right side of the track (>0) */
    tdble radiusl;      /**< Radius in meters of the left side of the track (>0) */
    tdble arc;          /**< Arc in rad of the curve (>0) */
    t3Dd center;        /**< Center of the curve */
    t3Dd vertex[4];     /**< Coord of the 4 corners of the segment.
                    <br>Index in:
                    - TR_SL
                    - TR_SL
                    - TR_EL
                    - TR_ER
                 */
#define TR_SL   0       /**< Start-Left corner */
#define TR_SR   1       /**< Start-Right corner */
#define TR_EL   2       /**< End-Left corner */
#define TR_ER   3       /**< End_Right corner */
```

Figure 19: Track information about track segment vertex and other parameters.

8.3 Real World Applications

The research work performed in this thesis could be used in numerous real world applications. Driving fast in curves causes a considerable number of accidents especially in mountainous areas where the road tracks are tricky and complex. This phenomenon is mostly controlled via differentials techniques using the car hardware and by the driver itself. A traction control system being part of a driver assistant system using advance AI techniques as we described in the section 5.3.6 could help in this matter. On foggy day, the visibility for the driver is less which sometimes leads to a number of accidents. In such a situation, the driver tends to drive slowly but despite this at busy intersections and bridges collisions are eminent and unavoidable. The overtaking agent defined in section 5.3.5 which makes use of various car and track sensors to avoid collision with the nearby cars could be a role model to deal with such a situation. Other potential applications are described as:

- A complete modular architecture for the driverless cars which could be further enhanced and modified for the real driverless cars if the complete sensor environment is provided.
- To test various car control solutions like traction control (different differential techniques could be tested), keeping safe speed limit in Curves and testing various ABS (Automatic Braking System) solutions etc.
- The Simulation Provides Pit shop which is used by the car controllers to repair damages. A strategy is used to stop there using available sensors. Similar strategy could be used to mimic parking assistant.

REFERENCES

[1] T. Uusitalo and S. J. Johansson, "A reactive mutli-agent approach to car driving using artificial potential fields," in *2011 IEEE Conference on Computational Intelligence and Games (CIG)*, 2011, pp. 203–210.

[2] "*TORCS*," *Wikipedia, the free encyclopedia*. 19-Oct-2012.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation. Proceedings*, 1985, vol. 2, pp. 500–505.

[4] "torcs." [Online]. Available: http://torcs.sourceforge.net/index.php?name=Sections&op=viewarticle&artid=1. [Accessed: 29-Jul-2013].

[5] L. Cardamone, D. Loiacono, and P.-L. Lanzi, "Learning drivers for TORCS through imitation using supervised methods," in *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009*, 2009, pp. 148–155.

[6] D. Loiacono, P.-L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. V. Butz, T. D. Lönneker, L. Cardamone, D. Perez, Y. Saez, M. Preuss, and J. Quadflieg, "The 2009 Simulated Car Racing Championship," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 2, pp. 131–147, 2010.

[7] J. Hagelbäck and S. J. Johansson, "Using multi-agent potential fields in real-time strategy games," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, Richland, SC, 2008, pp. 631–638.

[8] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-Heether, S. M. Lucas, M. Simmerson, D. Perez, R. G. Reynolds, and Y. Saez, "The WCCI 2008 simulated car racing competition," in *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, 2008, pp. 119–126.

[9] L. Cardamone, D. Loiacono, and P.-L. Lanzi, "On-line neuroevolution applied to The Open Racing Car Simulator," in *IEEE Congress on Evolutionary Computation, 2009. CEC '09*, 2009, pp. 2622–2629.

[10] E. Onieva, L. Cardamone, D. Loiacono, and P.-L. Lanzi, "Overtaking opponents with blocking strategies using fuzzy logic," in *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, 2010, pp. 123–130.

[11] M. V. Butz and T. D. Lönneker, "Optimized sensory-motor couplings plus strategy extensions for the TORCS car racing challenge," in *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009*, 2009, pp. 317–324.

[12] S. Priesterjahn, O. Kramer, A. Weimer, and A. Goebels, "Evolution of Human-Competitive Agents in Modern Computer Games," in *IEEE Congress on Evolutionary Computation, 2006. CEC 2006*, 2006, pp. 777–784.

[13] "Learning human-like Movement Behavior for Computer Games - Google Scholar." [Online]. Available: http://scholar.google.com/scholar?q=Learning+human-like+Movement+Behavior+for+Computer+Games&oe=utf8&um=1&ie=UTF-8&lr=&cites=5150172227815570812. [Accessed: 01-Jul-2013].

[14] "Eric J. Rossetter. A Potential Field Framework for Active Vehicle LanekeepingAssistance. Diss. 2003, Stanford University, 2004 - Google Scholar." [Online]. Available: http://scholar.google.se/scholar?q=Eric+J.+Rossetter.+A+Potential+Field+Framework+for+Active+Vehicle+LanekeepingAssistance.+Diss.+2003,+Stanford+University,+2004&oe=utf-8&rls=org.mozilla:en-US:official&client=firefox-a&channel=fflb&um=1&ie=UTF-8&lr=&cites=5884294687646582387. [Accessed: 01-Jul-2013].

[15] L. Cardamone, D. Loiacono, P.-L. Lanzi, and A. P. Bardelli, “Searching for the optimal racing line using genetic algorithms,” in *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, 2010, pp. 388–394.
[16] “TORCS Robot Tutorial.” [Online]. Available: http://www.berniw.org/tutorials/robot/. [Accessed: 07-Jul-2013].
[17] “Traction control system,” *Wikipedia, the free encyclopedia*. 21-Jul-2013.