PHYSICS SIMULATION FOR AN AUTOMOTIVE SIMULATOR

PHYSICS SIMULATION FOR AN AUTOMOTIVE SIMULATOR

Project Thesis

Neha Dike
Student Number: 20222291

University of Western Australia
School of Electrical, Electronic and Computer Engineering
Centre for Intelligent Information Processing Systems (CIIPS)

Supervisor: Associate Professor Thomas Bräunl

Submitted on: 31 October 2008

Neha Dike
78, Thomas Street,
Nedlands,
WA 6009.
31 Oct 2008.

The Dean
Faculty of Engineering Computing and Mathematics,
The University of Western Australia,
25 Stirling Highway,
CRAWLEY WA 6009.

Dear Sir,

I submit to you this dissertation ”Automotive Simulation” in partial fulfillment of the requirement of the award of Master of Engineering in Information and Communication Technology.

Yours faithfully,

.....
Neha Dike

Abstract

In recent years car manufacturers are have been looking at manufacturing fully autonomous private cars. This would not only make travelling easier but also help reduce car crashes and accidents. All this has lead to growing academic as well as industrial research in this field. However, the chances of human error always exist and can be very expensive to deal with in such applications. Hence, it is always a good idea to use simulators to test the software seperately before loading it on the corresponding hardware. Simulation provides the ability to experiment with test procedures that are dangerous and difficult to control, or to compute performance variables that are hard to measure experimentally. It enables engineers to perform planned tests in minimum time and cost.

The Automotive Simulator project has a server running the physics and a client modeling the graphics. The robots in the simulation are controlled by separate user programs which allow the user to interact with the simulation. The project aims at providing an open source simulator with an environment that can allow the user to simulate multiple vehicles in real time. This is an ongoing project involving a number of developers and I have contributed to improving the physics simulation, developing user programs for driving the robots and generating some test environments. Physics improvement majorly focuses on the addition of gears to the existing physics, with an aim of increasing realism of the vehicle behavior. User programs include providing a steering wheel interface for users driving the robot cars and developing an autonomous car control algorithm with the objective of generating traffic in the simulation environment.

Acknowledgements

I would firstly like to thank the University of Western Australia and specially the CIIPS group for giving me the opportunity of being a part of this project.

The project would not have been possible without the able supervision of Professor Thomas Bräunl. I thank him for giving me this learning opportunity and for his guidance throughout the duration of the project.

I would also like to thank Adrian Boeing for his help and direction that helped me in completing this project successfully. His knowledge and willingness to help have greatly benefited me.

Finally, I would like to thank all my project mates for their support and help in solving problems at various stages of the project.

TABLE OF CONTENTS

1. Introduction

This section gives a brief overview of the developments in the field of robot cars over the years, and points out the importance of using simulators for testing autonomous vehicles.

1.1 Background

The history of autonomous vehicles dates back to 1977 when they were driven on clearly marked roads free from obstacles, in Japan. Over the years, more and more enhancements have been made to these and today, autonomous vehicles are becoming cheaper as high speed computing has greatly minimised the need for complex and expensive hardware. In 2002, the Grand Challenge competitions were announced, sponsored by the Defense Advanced Research Projects Agency (DARPA), the central research organization of the United States Department of Defense. This was the first long distance competition for robot cars in the world. When the competitions first started, a number of international teams competed to drive fully autonomous vehicles through rough unpaved terrain, in a fairly obstacle free suburban setting. Later on, the complexity of the path and terrain was increased and in 2007 an urban environment was set up and cars had to navigate through traffic, on a 60-mile urban course within a six hour time limit while obeying traffic laws and avoiding accidents. [1]

Figure 1: The DARPA Grand Challenge. Photographs taken from [1]

For long, the motive for development of autonomous vehicles has been defense applications or space research purposes. However, this approach is changing

and car manufacturers are now looking at manufacturing fully autonomous private cars. This would help reduce car crashes and accidents. All this has lead to growing academic as well as industrial research in this field. However, the chances of human error always exist and these can be very expensive to deal with in such applications. Hence, it is always a good idea to use simulators to test the software seperately before loading it on the corresponding hardware.

Simulation provides us with the ability to experiment with test procedures that are dangerous to perform and difficult to control, or to compute performance variables that are difficult to measure experimentally. In case of vehicles, this problem is particularly relevant to rollover behavior. Test engineers can use driving simulation software to reduce test time and cost during development. Different test procedures can be designed to represent different road topologies and driving conditions. Well planned tests can be performed in a simulated environment prior to more expensive experimentation on the actual environment. [2] With simulation, the insight into the detailed features of the actual design is increased and the time between model changes is reduced as compared to the time for real tests. Hence, the need for automotive simulators.

The numbers of cars on roads all over the world are increasing daily and so are the accidents caused. Simulators are being widely used for driver training. Also, the amount of electronics in cars is increasing, with a number of driver assistance functions being provided. This increases complexity. Simulators can help test and improve these features. AutoSim can be used for any of these.

1.2 Automotive Simulator Project Objectives

The aim is to provide an open source simulator with an environment that can allow the user to simulate multiple vehicles in real time.

The main objectives of the project include:

- Providing different environments for the simulations, like residential areas, forest lands, urban settings etc. along with different weather and lighting conditions.

- Providing an environment with intelligent traffic and traffic lights.

- Simulating different types of vehicles.
- Allowing users to drive the vehicles in real time using an accelerator and brake pedal, gears and steering wheel controls.

- Allowing a number of clients to run simultaneously, thus controlling different vehicles.

- Making efficient use of data from the various sensors to make decisions about vehicle behavior.

- Making possible adaptive cruise control (ACC) as an extension to the conventional cruise control, where ACC keeps track of the allowed speed and ensures a given distance to the vehicles ahead by braking. The ACC could be seen as a step towards autonomous driving.

1.3 Thesis Structure

Chapter 2 discusses the literature survey that I had conducted at the beginning of the project. The topics covered include existing simulators, both open source and commercial ones, and information about the past developments of some important aspects related to automotive simulators such as navigation systems, map databases, vehicle dynamics and obstacle avoidance. Chapter 3 gives an overview of the AutoSim project; a brief description of the libraries used; an introduction to the main components of the project and includes screenshots to give a better idea of the project.

One of the main parts of the thesis, the physics simulation, is discussed in chapter 4. It includes an overview of the physics engine used; and a detailed description of the forces acting on a vehicle, aerodynamic effects and the gear functionality. Chapter 5 talks about the user interface available for interacting with the simulation and the control programs running behind the interface. Chapter 6 discusses autonomous driving, another important part of the thesis. It describes the control mechanisms and the driving algorithm used for autonomous control of robots in the simulation.

Chapter 7 discusses the tests and experiments carried out to verify the implementations, along with the results obtained and conclusions drawn from the results.

Finally, a conclusion is made in chapter 8 and future work necessary for improvements to the simulator is discussed in chapter 9.

2. Literature Review

Before actually starting work on the project, I started with a literarure review so as to have an understanding of work that was already done in this area.

A number of driving simulators already exist. They are being used widely for entertainment and for training of drivers. They are also used for research purposes by both universities and vehicle manufacturers. Universities carry out research on human factors, to monitor driver behavior, performance, and attention; and the vehicle industry research is aimed at designing and evaluating new vehicles or advanced driver assistance systems.

2.1 Existing Open Source Simulators

A number of open source simulators exist. Some of them are discussed below:

TORCS - The Open Race Car Simulator

TORCS is a multi platform car racing simulation system. It is used as AI racing game as well as a research platform. A joystick, mouse or a steering wheel can be used to control the cars. It provides around 20 tracks and graphic features like lighting, smoke, skid marks and glowing brake disks. However, as it is intended for racing, it does not provide an urban setting. Figure 2 shows screenshots of the TORCS simulator.

Figure 2: TORCS. Photographs taken from [56]

GRacer

It is a 3D simulator for motor sports, with five cars. Its physical simulation engine creates realistic drift, wheel spin, and accelerator turn. However, it has been designed from gaming point of view and has very limited tracks. Figure 3 shows a screenshot of the GRacer.

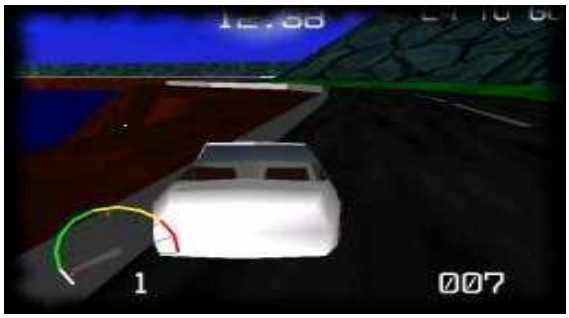


Figure 3: GRacer. Photograph taken from [57]

RARS - Robot Auto Racing Simulator

RARS is a competition for programmers and an on-going challenge for researchers in the field of Artificial Intelligence and real-time control. It consists of the physics simulation of cars racing on a track, a graphic simulation of the race, and a separate control program for each car. RARS is an open source initiative and all software and documentation is freely available. A number of cars and tracks are present. However, this is again a racing simulator and does not provide an urban environment. Figure 4 shows a screenshot of the RARS simulator.

Figure 4: RARS. Photograph taken from [58]

VDrift

It is a cross-platform, open source driving simulator. It was basically designed for drift racing with the aim of use in gaming applications, and provides support for mouse, joystick, game pad, steering wheel and keyboard controls. Its physics engine is based on the Vamos physics engine. [60]

Figure 5: VDrift Simulator. Photograph taken from [60]

Vamos:

Vamos is an automotive simulation framework with an emphasis on thorough physical modeling. It models most major systems of a car and includes a simulation of the engine, clutch, transmission and a limited-slip differential. Tires and suspension are also modeled. Figure 6 shows a screenshot of the Vamos based simulator

Figure 6: Vamos. Photograph taken from [59]

EyeSim:

The EyeSim simulator was developed at CIIPS at the University of Western Australia. It is a “multiple mobile robot simulator” that allows experimenting with the programs that run on the real robots. EyeSim includes simulation of the robot's

driving actuators and all the robot sensors; such as on-board vision, infra-red sensors, bumpers, odometers, PSD etc. [51]

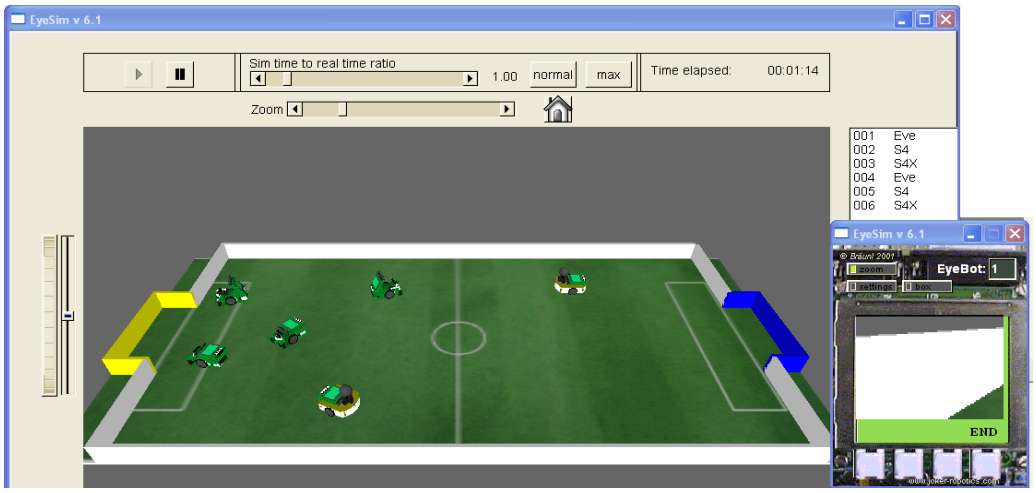


Figure 7: EyeSim. Photograph taken from [51]

SubSim

The SubSim is a simulation system for autonomous underwater vehicles, also developed at CIIPS at the University of Western Australia. It too provides a full physics simulation along with sensor and actuator simulation. SubSim uses the physics engine called Newton to model the motion of objects under the influence of forces and collisions. [50] AutoSim has benefited from the work done in SubSim; as there are a number of similarities in areas like sensor and API implementation.

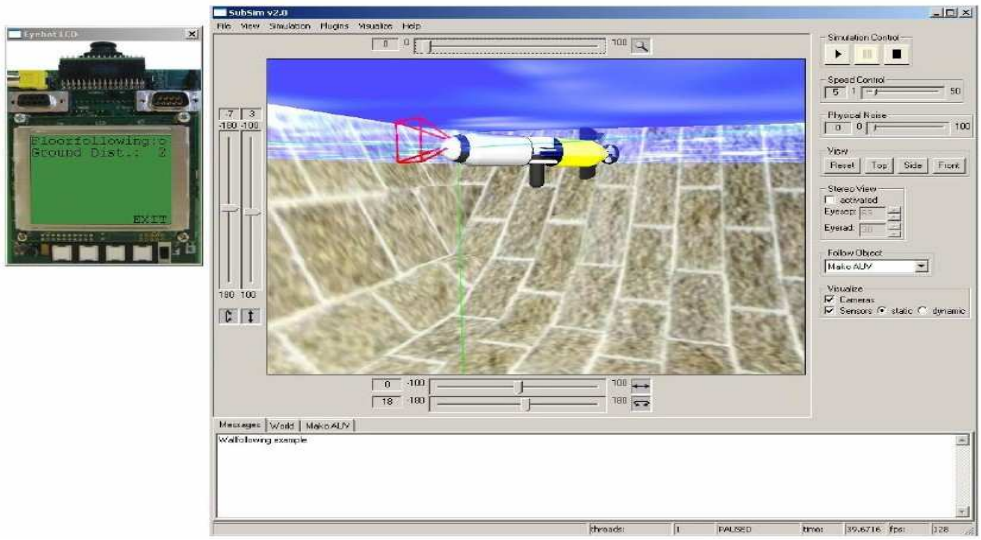


Figure 8: SubSim. Photograph taken from [50]

2.2 Driving Simulators used in the Manufacting Industry

A number of car manufacturers have come up with their own simulators for functionality testing, driver training and research purposes. Two of the famous ones are disscussed here.

2.2.1 The Honda Driving Simulator

In 2001, Honda introduced a driving simulator with a 6 axis motion base that could closely simulate vehicle dynamics in real world situations. Their aim was to provide new drivers with training on safe driving techniques.

This was the first time a 6-axis motion base and a seat with vibration were adopted for the education of general drivers. A cinemascopic projection device was used to provide a wide field of vision with high quality images. Image processing was applied to make buildings and other background appear more realistic. It was designed to enable training for both automatic as well as manual gear driven vehicles. [54]

Figure 9: Honda Driving Simulator. Photograph taken from [54]

2.2.2 The Toyota Driving Simulator

In December 2007, Toyota revealed a huge driving simulator that was designed to provide realistic driving environments, to study driving habits and for verifying the effectiveness of safety technology with the long term goal of designing cars that would never crash.

The simulator consists of a real car positioned on a mechanical platform inside a 7.1 meter dome. A tilt mechanism, vibration apparatus and other devices have been used to manipulate the dome as the driver operates the vehicle. The dome itself, acts as a giant 360-degree video screen which can be moved by means of computer control. With the aim of simulating a real driving experience, it includes sound effects, and the moving screen gives a sense of speed, acceleration and riding comfort.

Figure 10: Toyota driving Simulator. Photograph taken from [55]

Figure 11: Toyota Simulator 360 degree video screen. Photograph taken from [55]

In their paper, 'Development of Universal Driving Simulator with Interactive Traffic Environment', the authors have reported their concept and progress in developing driver model of the motion based driving simulator with interactive traffic environment. It gives a good idea of the various considerations that need to be taken into account while designing such a simulator. [11]

2.3 Navigation Systems

The first prototype of a full functional car navigation system, called EVA, was presented by Bosch in 1985. There after, a lot of work has been concentrated on car navigation systems.

The main role of a car navigation system is to find the car position as precisely as possible. Initially, most car navigation systems estimated the car position from dead reckoning and the Global Positioning System (GPS). However, because of the unknown GPS noise, the estimated position has an error. To solve this problem, a map matching method was introduced, which uses a digital road map to correct the position error. Sinn Kim and Jong-Hwan Kim have proposed an adaptive fuzzy network based map matching algorithm which can find the exact road on which a car moves. [8]

In January 2007, Dragan Obradovic, Henning Lenz, and Markus Schupfner, in their paper, presented two novel sensors and information source fusion methods implemented in the Siemens car navigation systems. This navigation system which implements a Kalman Filter based approach and pattern recognition for updating the drive trajectory, is a standard commercial product of Siemens AG and has been implemented in Opel, Porsche, Alfa Romeo, and Lancia models since 2000. [9]

2.4 Map Databases

In 1984, members of the Ministry of Construction, private companies, and academic institutions in Japan agreed that digital road map database using standardized specifications must be constructed; and major companies in the automobile, electrical machinery and instrumentation, mapping and surveying industries and the financial services sector formed Japan Digital Road Map Association (JDRMA) in August 1988. A large number of private companies continue to edit the database information for use in their car navigation. [5]

In the following few years, mapping of major parts of Europe was also achieved. We now have readily available map databases like google map, open street map etc. We need to choose a database that provides a free service. We need to be aware that most maps we think of as free actually have legal or technical restrictions on their use, which can holding back people from using them in their own creative applications. An open source simulator requires a completely free map database.

Open Street Map

Open Street Map (OSM) [15] provides free maps of the whole world. Anybody can upload a map to the OSM database, download a map from the database or edit an already existing map. Open Street Map allows a user to view, edit as well as redistribute geographical data from anywhere on the Earth. The Java OSM editor (JOSM) can be used for viewing and editing the maps downloaded from the open street map database. The details of using OSM files are explained in a tutorial in [38].

Google Maps

Google Maps provides high resolution satellite images of streets in most urban areas of the world. These maps have been made available for personal, non-commercial use only. The copyright clearly specifies that for commercial users, Google Maps should be restricted to internal use only and that redistribution would be considered illegal. [62]

2.5 Environment Construction

In 2003, researchers in Paris came up with a method to develop a system, which is capable of acquiring data in three dimensions for building an accurate 3D model of an urban outdoor environment. It was a vehicle borne laser scanner with a GPS plus INS plus odometer based navigation system. The system could operate in real time and build high resolution maps of static environments around the vehicle. [4]

In 2006, researchers in Japan proposed a simple and useful method of making a 3D geographical map by mapping onto a given 2D map, with the data measured on the streets using video cameras and several sensors on a car. They confirmed that it was possible to allocate in a 2D map, the road surface data that are difficult to collect using aerial or satellite photograph resources.[6]

Also in 2006, researchers at the Guangdong University of Technology, China, proposed a virtual traffic environment model. They used the 3dsMAX software to construct 3D entity, then transform to .3ds format and import to Eon Studio for simulation. The finished virtual traffic environment model includes different kinds of road, such as city road, mountain road, village road. It also can reflect the time changes during a day and weather variations. Far scene and close by

scene could be differentiated and the vehicles showed intelligent behavior, such as avoiding collision and overtaking other cars. [14]

Figure 12: Typical simulator environment. Photograph taken from [14]

2.6 Vehicle Dynamics

In June 2001, Andy R. W. Huang and Chihsiuh Chen constructed a five axis driving simulator based on vehicle dynamics analysis and system integration. MATLAB was used to provide files for mathematical calculations and was chosen to ease the burden of writing numerical analysis codes in C++. However, since C++ Builder makes available many GUI design functions superior to MATLAB, MIDEVA was used to transfer the m-files into C++ codes. The constructed five axis motion platform could perform lateral and vertical displacements and roll, pitch, and yaw motions similar to the 5 DOF motions of a road vehicle. [12]

Later in 2005, at the Center for Collaborative Research, the University of Tokyo, an experiment platform was developed for studying drivers' characteristics on the motion based driving simulator which was equipped with a body cabin, a 360 degrees field of view display system, a steering wheel with an AC servo motor, surround audio system and 6 DOF motion system with turning table. [11]

2.7 Obstacle Avoidance

In 2002, researchers at the University of Technology, Petronas came up with a fuzzy logic approach of reactive navigation for obstacle avoidance. The fuzzy logic controllers used a set of 50 rules to decide the steering angle and velocity depending on the reactive inputs of distance of obstacle and angle of obstacle.[3]

In [10], automotive crash simulation has been considered in a stochastic context. Since crash is a non repeatable phenomenon, decision regarding crash effects cannot be based on a single test, and should be replaced by stochastic simulation. In parallel crash simulation, variations of the calculated results can be observed; and in many cases, even for the same input data set, we can get different results. Data mining algorithms can be applied to describe this kind of behavior.

3. Overview of AutoSim: The Automotive Simulation System

Basically AutoSim is an open source simulator. The aim is to provide an open source simulator with an urban environment. Such an environment is cluttered with moving obstacles. Motion safety becomes a critical issue in this case. A complete simulator should be able to provide an environment consisting of different lighting conditions, weather conditions, vehicle types and even pedestrians and cyclists.

An open source simulator requires a completely free map database; that is, one that not only allows you to use the map but also to redistribute it. Geographic data from the Open Street Map project [15] that supplies information about streets, buildings and land use has been used to build the world used for simulation. All details of the objects in the simulated world, their positions and the maps themselves are stored as XML files and can be easily edited by changing the necessary parameters in the corresponding files as they are based on a standard format. [36]

3.1 Libraries Used

A number of freely available libraries have been used in the project. This helps to introduce modularity along with saving the effort and time required for developing and testing. Some of the important ones are PAL, RakNet, SDL, Bullet, Qt and Irrlicht.

The Physics Abstraction Layer (PAL) [20] provides an interface to a number of different physics engines thus enabling the use of multiple physics engines within one application. Amongst all the physics engines supported by PAL, bullet has been chosen for this project. Bullet is a *"professional open source multithreaded 3D Collision Detection and Rigid Body Dynamics Library"*. [16] Both PAL and Bullet are further described in section 4.

Raknet is a cross-platform C++ networking engine. It has been designed to be a "*high performance, easy to integrate, and complete solution*" for games and other applications. [18] Although it could be used for any networked application, it was developed specifically for rapid development of online games and the addition

of multiple players to single player games. RakNet allows any application to communicate with other applications on the same computer, over a LAN, or over the internet.

SDL or the Simple Direct Media Layer is a cross-platform multimedia library used for providing low level access to the user interface devices like keyboard, mouse, joystick and the steering wheel. SDL is written in C, but works with C++ and has bindings to several other languages. It is distributed under a license that allows you to use SDL freely in commercial programs. [13]

TinyXml is an open source xml parser for the C++ language. It parses the XML into a DOM like tree. It is a available for free, is a well documented software and is easy to integrate into programs. It can both read and write XML files. However, TinyXml does not process DTDs and has no facility for handling XML namespaces. [17]

The Irrlicht Engine is an "*open source, cross-platform, high performance, real-time 3D engine written in C++*". It is a high level API suitable for creating complete 2D and 3D gaming applications and visualizations. It is well documented and is easy to use. It includes features for visual representations of dynamic shadows, particle systems, character animation, and collision detection. [19]

Qt is a cross-platform application development framework, widely used for the development of GUI based applications, and sometimes also used for developing non GUI programs such as console tools and servers. It allows you to develop applications and user interfaces once, and deploy them across many desktop and embedded operating systems without having to rewrite the source code. Qt includes a rich C++ class library and an API, integrated tools for GUI development, and support for Java and C++ development. [39]

3.2 Project Components

The project basically consists of a server, a client, user programs, an osm manipulator program and an installer.

The project has a server running the physics and a client modeling the graphics. This makes it possible to have a number of clients. The user interface on the client and server provides a number of selection options to the users. These include selection of the world file, selection of the robot, selection of the user program, variable wind speed and different viewing positions. Apart from that, separate batch files are available to run the simulation for different combinations of

worlds, robots and user programs. Wind speed and the viewing position can be changed while the simulation is running. A speedometer is available for the user to check speed. The reset button can be used to put the vehicle being driven in to start position. This is useful specially while testing, when the vehicle may turn over or reach out of the bounds of the defined world. Figure 13 shows the client side user interface and figure 14 shows the server side user interface.

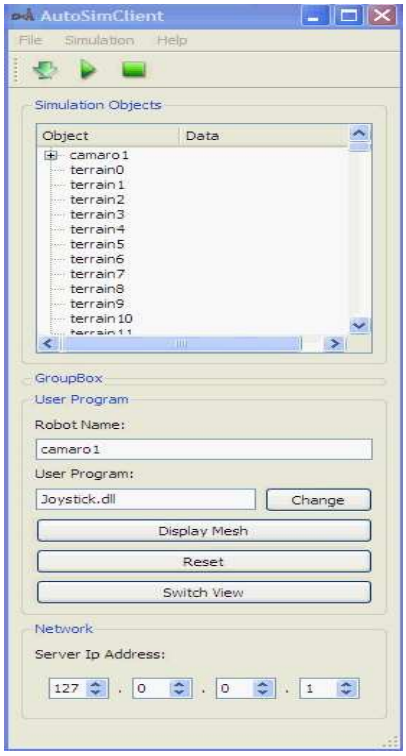


Figure 13: AutoSim Client interface

The robots in the simulation are controlled by separate user programs which allow the user to interact with the simulation using various input devices like the steering wheel, a joystick or the mouse. These programs use the robot's virtual sensors to get data from the environment, make decisions depending on user inputs and some predefined criteria, and send appropriate signals to the actuators. The robot specifications and configurations are also stored in XML files, making it simple to add devices to it; e.g. gears or lights to the chassis of a car. Note that, vehicles are not the only robots. Other objects that can change state are also specified as robots. Currently these include only traffic lights; however, in the future other robots may be added such as pedestrians, cyclists etc.

The OSM manipulator is a program which enables adding houses to an osm world file in an easy manner. Basically, it is used for creating a new world or modifying an existing one. It has been explained in detail in [37]

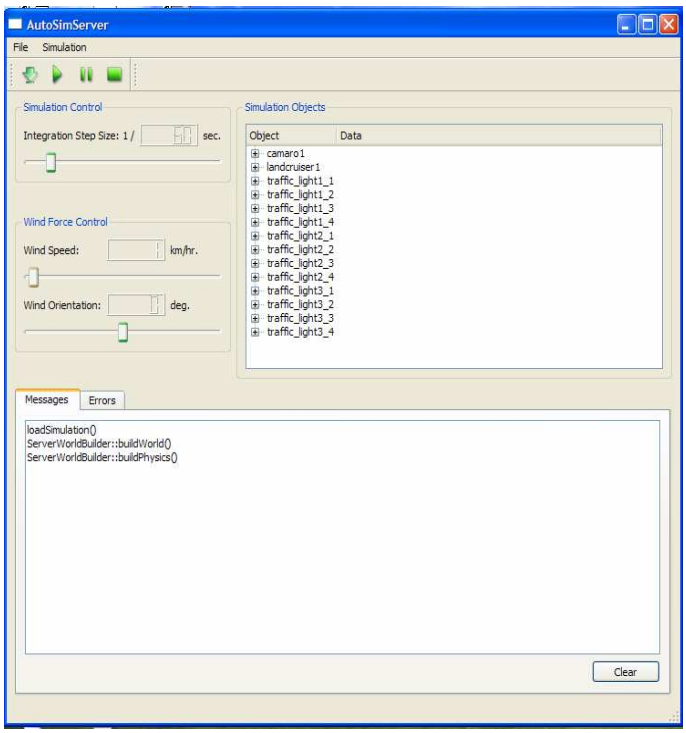


Figure 14: AutoSim Server Interface

Figures 15 and 16 show the actual driving environment from the driver's view and the third persons view respectively. These are the two views of the actual driving window.

Figure 15: Driver's view of the AutoSim environment

Figure 16: Third Person view of the AutoSim environment

4. AutoSim Physics Simulations

4.1 Physics Engines

Wikipedea has defined a physics engine as a "*computer program (software component) that uses variables such as mass, velocity, friction and wind resistance to simulate and predict effects under different conditions that would approximate what happens to a vehicle in either real life or a fantasy world*".[61] A number of physics engines are available today. Some of the open source ones are Ageia, Bullet, Chrono Engine, DynaMechs, DynoMo, FastCar, Newton, ODE and many more.

4.1.1 PAL (Physics Abstraction Layer):

The physics abstraction layer was developed at the University of Western Australia to provide a common interface to a number of different physics engines, thus enabling an easy use of multiple physics engines within the same application. Basically, upgrading to a new physics engine is easy as PAL does not restrict you to one particular engine. PAL provides an extensive set of features which include simulating different devices or loading physics configurations from common file formats like XML or COLLADA, and hence is easy to use. It provides an interface for a number of sensors and actuators. The sensors include Contact sensor, Compass sensor for angular position, GPS (Global Positioning System) sensor, Gyroscope for angular velocity, Inclinometer for angular position, PSD (Position Sensitive Device) sensors, Velocity meter for linear velocity, and Transponder to get the distance between two objects. The actuators available include Force actuator, DC Motor, Servo Motor, Hydrofoil, Propeller and Spring. [20]

Some of the physics engines supported by PAL are Ageia PhysX, Box2D, Bullet, IBDS, JigLib, Newton, ODE, OpenTissue, Simple Physics Engine, Tokamak, and TrueAxis. Of these, AutoSim currently uses Bullet.

4.1.2 Bullet Physics

Bullet [16] is an open source software multi-threaded 3D Collision Detection and Rigid Body Dynamics Library and is free for commercial use. Its main features include continuous Collision Detection and a modular approach. It implements a fast collision detection algorithm and the collision shapes include sphere, box, cylinder, cone, convex hull, and triangle mesh. Bullet provides rigid body dynamics for the Blender 3-D modeling, rendering, and animation package. It provides support for convex and concave meshes and provides 6 degrees of freedom for hinge, etc.

4.2 Vehicle Physics

4.2.1 Forces

The forces acting on a vehicle can be divided into two types, namely, longitudinal and latitudinal. Longitudinal forces act in or against the direction of motion of the vehicle body. These include engine force, braking force, rolling resistance and drag or air resistance. Together these forces control the acceleration or deceleration of the car and therefore the speed of the car. Lateral forces allow the car to turn and arise as a result of tire deformation. These forces are caused by sideways friction on the wheels.

The total longitudinal force is the vector sum given as [40]:

$$F_{long} = F_{drive} + F_{drag} + F_{rr}$$

F_{drive} is the engine force given as:

F_{drive} = u * Engine force,
where u is a unit vector in the direction of the car's heading,

The engine force calculation has been discussed in section 4.4.4.

In the case when brakes are applies, the drive force is replaced by a braking force which is oriented in the opposite direction, thus applying a resultant negative force.

The other two forces, which are the resistance forces, act in the opposite direction. At low speeds the rolling resistance is the main resistance force and at high speeds the drag takes over.

The force of air resistance is given as [40]:

$F_{drag} = - C_{drag} * v * |v|$
Where C_{drag} is a constant and
v is the velocity vector and the notation |v| refers to the magnitude of vector v.

It is discussed in further detail in the next section.

Rolling resistance is caused by friction between the rubber and road surface as the wheels roll along and friction in the axles, etc. It is given as [40]:

$F_{rr} = - C_{rr} * v$

where C_{rr} is a constant and
v is the velocity vector.

The following chart from [40] gives a good idea about these forces. In this diagram the X-axis denotes car velocity in meters per second and the Y-axis denotes force values. The engine force has been set to an arbitrary value.

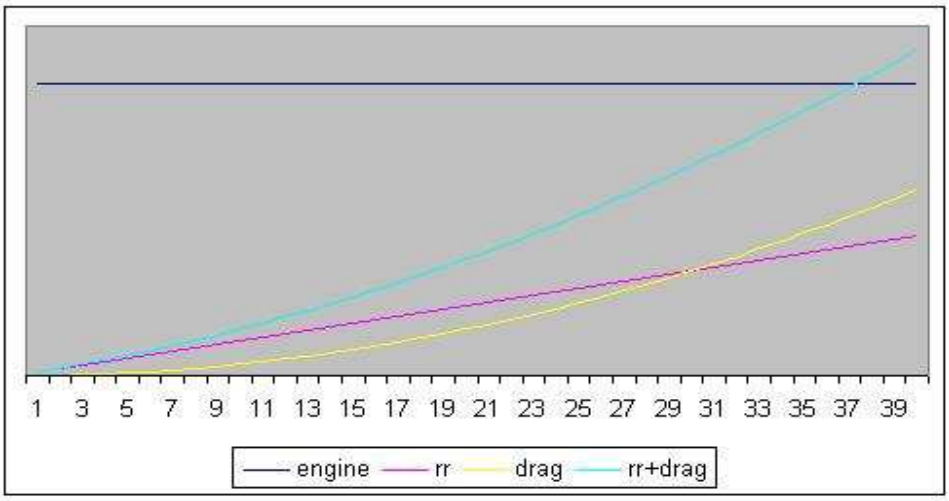


Figure 17: Longitudinal forces on a vehicle. Figure from [40]

Weight transfer

Balancing a car is nothing but controlling its weight transfer. Inertia and adhesive forces give rise to weight transfer through Newton's laws. Figure 18 from [41] shows the balancing forces.

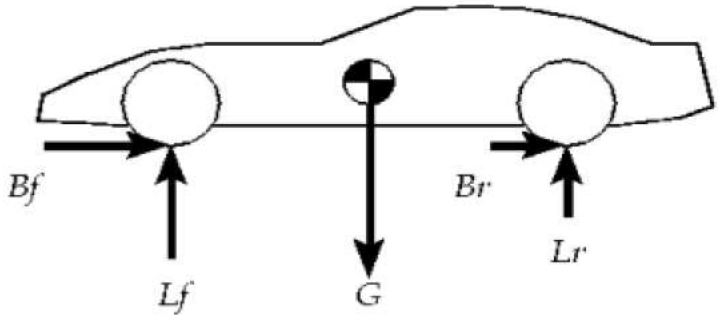


Figure 18: Balancing Forces on a vehicle. Figure from [41]

G is the force of gravity that pulls the car towards the centre of the Earth. This is due to the weight of the car.
L_f is the lift force exerted by the ground on the front tire, and L_r is the lift force on the rear tire.
B_f and B_r are the braking forces on the front and rear tires respectively. These will be replaced by the corresponding drive forces in case of acceleration.
The imbalances between the front and rear lift forces account for understeer or oversteer conditions. An under steering situation is where a car is not turning as much as desired, and oversteer means the car is steering too much.

4.2.2 Physics of Turning Vehicles

Physics of turning at low speed is different from the physics of turning at high speed. At low speeds, the wheels roll in the direction they're pointed at.

At higher speeds, the inertia effect comes into picture and the wheels can be heading in one direction while their movement is still in another direction. Pushing a wheel sideways is very difficult and causes a lot of friction force. In high speed turns, wheels are being pushed sideways and we need to take these lateral forces, known as the cornering force, into account. This force depends on the slip angle (alpha), which is the angle between the tire's heading and its direction of travel. As the slip angle grows, the cornering force also grows. At low slip angles, the relationship between slip angle and cornering force is linear, in other words;

$F_{lateral} = C_a * alpha$

where the constant C_a is known as the cornering stiffness

For better approximation of the relationship between slip angle and lateral force, the Pacejka Magic Formula [44] is used.

It is important to remember that in case of front wheel driven vehicles, the steering angle is the angle that the front wheels make with respect to the orientation of the car. There is no steering angle for the rear wheels; which are always in line with the car body orientation.

4.2.3 Physics of Rollover Accidents

When a vehicle turns, it still wants to go straight due to inertia, so the tires must create a lateral force as discussed above. This force is the centripetal force. When a vehicle is traveling at speed, and a sudden and extreme side turn is made, the centripetal force can be enough to cause a rollover.

We know that all the weight of the vehicle acts as though it is at a point at the height of the center of gravity. The centripetal force being applied by the tires and the road is at the height of the road surface. This horizontal force does not act directly at the center of gravity because it is at the wrong height. Therefore, it not only acts to push the weight (center of gravity) sideways but it also acts in a way that tends to twist the vehicle around the center of gravity.

Turning the front wheels tends to create a lateral force (f) by the ground on the tires. This causes an acceleration = f/m which depends on the speed and radius of the turn. This force f for a circular motion is given as: [40]

$$f = (w * v * v) / (g * r).$$

Where,
w = vehicle weight;
v = vehicle velocity;
r = circle radius;
g = acceleration due to gravity.

That's the sideways force that has to be exerted on the vehicle, by the tires' grip on the road, in order for it to go around that-sized circle at that speed.

Figure 19 shows the force diagram for a turning vehicle.

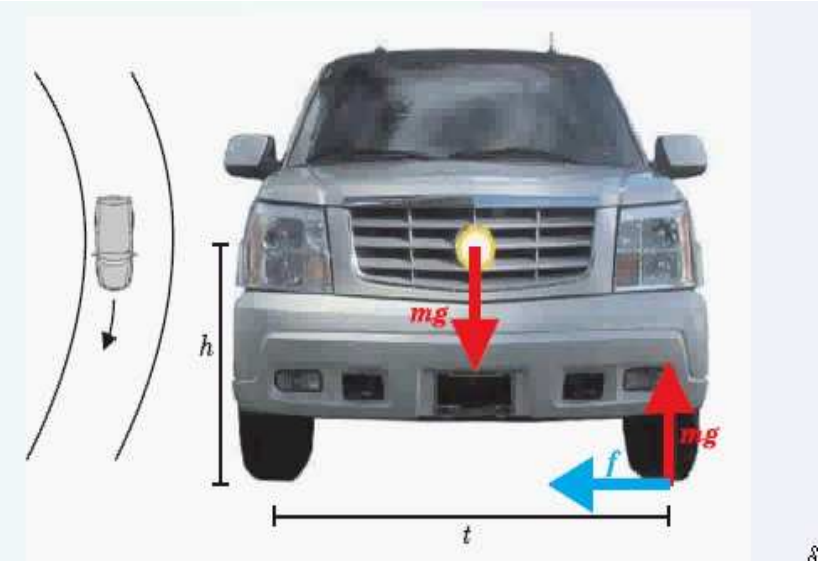


Figure 19: Forces during a turn. Figure from [45]

This force would have to lift up the weight of the vehicle to roll it over. On a flat road, the vehicle will tend to roll towards the outside. In other words, turning causes the load of the car to shift toward the two tires on the outside of the turn. The condition for the inner wheels to lift off the ground is given as: [45]

$$a / g = t / 2h$$

Where,
t is the distance between the tires or the track width,
h is the height of the center of gravity,
g is the acceleration due to gravity.

t / 2h is called the vehicles' static stability factor and is used to determine the rollover resistance ratings of vehicles. In [49], the author has given a detailed explanation of the development of the static stability factor.

Thus, it can be noted that, the higher a vehicle's center of gravity and the narrower its track width the less stable it will be during turnings. Following this, one can say that it is not surprising that SUVs are more prone to rollover than other vehicle types. They have a higher ground clearance and a narrow width which results in a higher Static Stability Factor than passenger cars, indicating a greater tendency to roll over.

4.3 Aerodynamic effects

Aerodynamics is the study of the relative motion of a solid body with respect to the surrounding air. Aerodynamic actions in vehicles are due to the relative motion of the vehicle and the air. Wind velocity and direction, vehicle speed and direction and shape of the vehicle are the important factors in determining aerodynamic effects. The force of a high speed wind can blow a vehicle over or cause it to slide sideways, depending on the direction of the wind and the friction coefficient. In [22], the author has explained with equations the cases of turning over and sliding sideways with conditions that cause each of these phenomenon.

In the presence of wind, a car is subjected to 3 forces; namely, drag, drift and lift. The drag force exists along the axis of the vehicle motion and acts as a resistance to the vehicle's motion. It needs to be taken into consideration when energy saving is important. In vehicle design, the aim is to reduce it as much as possible. The other two components determine the vehicle's stability and its sensitiveness to gusty side winds. [25] Drift is the force acting sideways on the vehicle and lift is the vertical component.

The figure 20 below, shows the coordinate system and the forces and moments along the different axes, with origin located at centre of gravity.

.

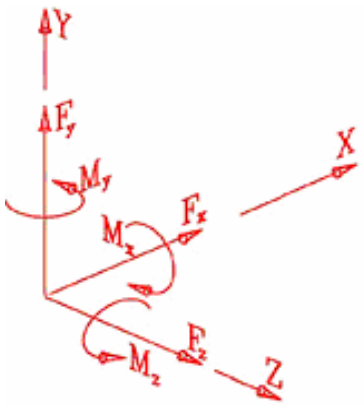


Figure 20: Forces and moments with respect to the axes of the coordinate system. Figure from [21]

Forces and moments depend on the square of the wind's velocity. They are given by the following formulae [21]:

$Fx = \frac{1}{2} * \rho * C_{Fx} * A * V_x^2$ ---- drift force;

$Fy = \frac{1}{2} * \rho * C_{Fy} * A * V_y^2$ ----- lift force;

$Fz = \frac{1}{2} * \rho * C_{Fz} * A * V_z^2$ ----- drag force;

$Mx = \frac{1}{2} * \rho * C_{Mx} * A * V_x^2 * h$ --- pitching moment;

$My = \frac{1}{2} * \rho * C_{My} * A * V_y^2 * h$ --- yawing moment;

$Mz = \frac{1}{2} * \rho * C_{Mz} * A * V_z^2 * h$ --- rolling moment;

Where,
ρ is the density of air;
A is the frontal area of the car;
$V = [V_x, V_y, V_z]$ is the wind velocity;
h is the height of the centre of gravity above the surface of the road;
C_F is the coefficient
(C_{Fx} is the coefficient of drag, C_{Fy} is the coefficient of lift, C_{Fz} is the coefficient of drift, C_{Mx} is the coefficient of roll, C_{My} is the coefficient of yaw and C_{Mz} is the coefficient of pitch.)

For practical purposes it can be assumed that the wind is acting in the horizontal xz-plane and the lift force is negligible.

In principle, all of these forces influence a vehicle's dynamics. Traditionally, test engineers distinguish between a vehicle's behavior in still air as handling, and its behavior in the presence of a crosswind as crosswind sensitivity. [24]

4.3.1 Handling

For a vehicle moving through still air, the air flow is almost symmetric about the vehicle's plane of symmetry. Lift, pitching moment and drag are therefore the only aerodynamic components that come into picture here. The vertical force tends to lift the vehicle. The pitching moment changes the load distribution between the front and rear axles, which alters the steering properties of a vehicle. With increasing speed, a negative pitching moment introduces a tendency to over steer. But, these effects are hardly noticeable at normal wind speeds. At normal driving speeds, these effects are negligible for most vehicles, but they become important in case of race cars.

4.3.2 Crosswinds

In a crosswind, the air flow around a vehicle becomes asymmetric and so a side force, a yawing moment, and a rolling moment are produced. At the same time, drag, lift, and pitching moment are altered, and normally they are increased. Of these, yawing moment and side force are the significant components when studying crosswind behavior of a vehicle. The yawing moment tends to twist the vehicle away from the wind and is the most important factor affecting directional stability of the vehicle. The increased angle of yaw, the yawing moment, and the side force cause instability. Simulators can be made to change any of these parameters independently of each other to study the driver's reaction in the event of sudden crosswind gusts, and of course to estimate the stability of the vehicle.

Note that effects similar to crosswinds are observed while passing another vehicle in still air. Similarly, vehicles passing behind a bridge tower in strong cross winds are affected by sudden change of wind forces acting on them, which may cause accidents due to miss-steering by the drivers. [27]

In [21], parameters such as wind velocity and direction; frictional coefficient; forces of inertia; gravitational pull; camber of the road; vehicle speed and damping and elastic forces are studied to estimate vehicle stability, safety, turnover characteristics and wind-related traffic accidents.

It is important to remember that, the effects of handling and especially crosswinds need to be studied differently and in more detail for race cars. In fact, study of aerodynamics is greatly responsible for the success of race cars. In [26], the author has discussed the details of race car aerodynamics.

4.3.3 Wind Effect Simulation in AutoSim

As an attempt to enable the use of AutoSim for testing vehicle behavior under exceptional weather conditions, AutoSim simulates the effect of strong gusts of wind. Sliders are provided for users to vary the wind speed and the wind direction; and accordingly the drag and drift forces acting on the vehicle in the simulation are calculated. It is assumed that the lift force is negligible. At very high wind speeds, the vehicle can be seen moving backwards, forwards, or shaking sideways towards either side; depending on the direction of wind selected. The wind direction has been set in terms of angle with respect to the axis parallel to the length of the vehicle and passing through the center of mass of the vehicle; and it can be varied to a precision of one degree.

The maximum wind speed available is 300 km/h. For reference, the speed in a typical strong storm is above 200 km/h. [53]

4.3.4 Dust Behavior

Another rare, but interesting area related to this topic is the study of the impact of dust generated by a traveling vehicle. Dust is caused by different factors such as natural wind, a fast traveling vehicle and an unpaved road. In [28], the authors have described a method for simulating the dust behavior caused by a fast traveling vehicle in real time. Though dust has not been considered in AutoSim, it can be simulated in the future.

4.4 Gear Function

A gear is basically a wheel with teeth along its rim and is used to transmit power from one shaft to another. In vehicles, a combination of two or more gears is used to change the rate of wheel rotation and the amount of torque at the wheel. Torque is a force that tends to rotate or turn the car's axle. The more the torque, the faster the car will initially accelerate. But, there is a trade off between torque, the amount of rotational force, and the speed at which the wheels of the car will turn. In other words, torque and top speed have an inverse relationship.

4.4.1 Gear Theory

Automobile drive trains generally have two types of gearing; namely the transmission gears and the differential gear. The transmission gear contains a number of different sets of gearing that can be changed to allow a wide range of vehicle speeds, and the differential gear contains one set of gearing that provides a force magnification at the wheels. [29]

Transmission uses gears to make more effective use of the engine's torque, and to keep the engine operating at an appropriate speed. The function of any transmission is to transfer engine power to the driveshaft and rear wheels in a rear wheel driven vehicle or axle shafts and front wheels in a front wheel driven vehicle. Gears inside the transmission change the vehicle's drive wheel speed and torque in relation to engine speed and torque. Lower gear ratios; that is, numerically higher

gears act as torque multipliers and help the engine to develop enough power to accelerate from a state of rest. [33]

Gear Ratios

Wikipedea has defined gear ratio as "*the relationship between the numbers of teeth on two gears that are meshed*". [63] It can be given as:

Gear ratio = No. of teeth in input gear / No. of teeth in output gear

Automatic and manual gears

There are two types of transmissions in road vehicles: the manual gear transmission and the automatic transmission. [31]

The important difference between a manual and an automatic transmission is that in manual transmission different sets of gears are locked and unlocked to the output shaft to achieve the different gear ratios, while in an automatic transmission; the same set of gears are used to achieve all of the different gear ratios. The mechanism used to achieve this in an automatic transmission is based on the use of planetary gearsets. [32]

4.4.2 Manual Transmission

Traditionally, vehicles have been designed with a gear shaft and a clutch pedal to enable the drivers to change gears at their discretion. The manual transmissions where a driver must shift from gear to gear preceded the automatics by several decades. A manual gear transmission system comprises of a gearbox, a clutch, a propeller shaft and a drive axle with a differential.

The main aim is to be able to achieve a maximum speed with the available engine. Proper gear selection also ensures that the characteristics of the engine are properly exploited to achieve maximum fuel efficiency.

Sequential Manual Transmission

A sequential manual transmission or a sequential manual gearbox is a type of manual transmission where gears have to be selected in a fixed order, rather than providing direct access to specific gears. Sequential manual transmissions will allow

the driver to select only the gear directly before or after the currently engaged one. This allows shifting between gears without the use of a clutch.

In traditional manual transmissions, the driver can move from any gear to any other gear, by moving the gear shift shaft to the appropriate position. This type of transmission is also referred to as an H-pattern transmission because the path that the shaft takes as it selects the various gears is an H manner. In this case, the clutch must be disengaged each time the new gear is selected; in order to disengage the running engine from the transmission and stop all torque transfer. [52]

4.4.3 Automatic Transmission

An automatic transmission is an automobile gearbox that can change gear ratios automatically as the vehicle moves, thus freeing the driver from having to shift gears manually. However, most automatic transmissions offer the driver a certain amount of manual control over the transmission's shifts apart from the selection of forward, reverse, or neutral gears.

Automatic transmission usually consists of a torque converter and an automatic gear box. Automatic transmissions are based on planetary gear sets. They use some arrangement of one or two central sun gears, and a ring gear, with differing arrangements of planet gears that surround the sun and mesh with the ring. Automatic transmissions may be either hydraulically controlled (using pressure sensors) or electronically controlled (using electronic sensors).

In most road vehicles, the transmission will still up shift automatically into the next higher gear ratio if the engine reaches its maximum permissible speed in the given range and downshift when the throttle is fully open, provided the current revs of the engine can accommodate a higher rpm as a result of a downshift. The more advanced electronically controlled transmissions provide additional features like downshift automatically when going downhill to control speed and reduce wear on the brakes; upshift when braking on a slippery surface to reduce the braking torque applied by the engine; and prevent upshift during turnings. [32]

In [34], the authors have discussed a fuzzy logic based approach for implementing automatic gear transmission.

4.4.4 Gear Physics Equations

We basically need to come up with a means to calculate the acceleration that will be applied, based on the torque generated.

By Newton's law, force is given as F = m a

Hence, acceleration a = F / m

Where,
m is the mass of the vehicle
F is the force on the car.

Here, we consider only the drive force, amongst all the force components discussed in section 4.2.1, since all other forces have been taken care of by the physics engine.

The drive force is given as: [38]

$F_{drive} = u * T_{drive} / R_{wheel}$

Where,
u is the unit vector in direction of vehicle,
R_{wheel} is the radius of the wheel and
T_{drive} is the drive torque.

The drive torque is given as [38]

$T_{drive} = T_{engine} * r_g * r_d * n,$

Where,
T_{engine} is the actual torque produced by the engine at a given RPM
r_g is the gear ratio
r_d is the differential ratio
n is transmission efficiency

In the above equation, the gear ratio, differential ratio and transmission efficiency are fixed for a given vehicle, and we need to calculate the engine torque.

For calculating the engine torque, we first need to find the rpm generated.
RPM can be calculated as: [35]

rpm = (mph x final gear ratio x 336) / tire diameter

mph is the speed in miles per hour.

An automobile uses gear ratios in both the transmission and the drive axle to multiply power. The two ratios multiplied together equal the final drive ratio. Tire diameter also has an effect on a vehicle's final drive ratio. As tire diameter changes, so will engine rpm at a given speed.

Hence,
rpm =(velocity in miles per hour)* gearRatio* diffRatio * 336 / (2 * wheelRadius)

The plot below shows the typical relationship between rpm of a vehicle and the engine torque.

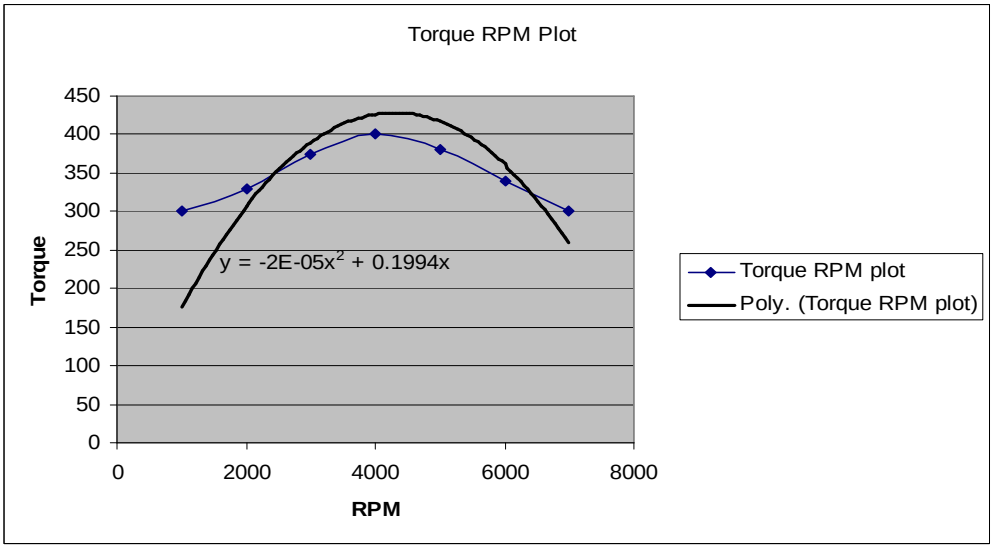


Figure 21: A typical torque RPM curve

Thus, the engine torque can be obtained from the rpm value calculated. The rpm of a vehicle for different gear ratios has been obtained, and hence the drive force required.

4.4.5 Gear Implementation in AutoSim

The gear functionality has been implemented in AutoSim based on the above set of equations.

From the above analysis, it can be seen that, there are a number of constants, specific to a particular vehicle. They have been listed below:

- Vehicle Mass
- Wheel radius
- Total number of available gears
- Gear ratio corresponding to each gear number
- Differential ratio
- Transmission efficiency
- Gear mode: auto, manual

All these parameters have been added to the xml for each of the robot cars under a new tag <gearDevice>. The procedure for adding devices to a robot has been explained in the appendix.

The rest of the parameters have been calculated based on the driver's inputs to the vehicle/robot.

Currently, it has been assumed that all vehicles have five gears in addition to a neutral. However, the number of gears and the corresponding ratios can easily be changed by manipulating the xml files.

In the implementation of manual gears, AutoSim implements the sequential manual transmission, which is best suited for the available hardware. One shift lever; namely the one on left hand side, is pulled to select the adjacent higher gear and another shift lever on the right hand side is used to select the adjacent lower gear. It is shown in figure 22.

Also, maximum and minimum limits have been set for rpm values that can be attained.

In case of manual gears, when either limit is reached, the rpm value remains at that value irrespective of the acceleration, until acceleration is applied in the opposite direction to change the rpm value. For instance, once the maximum value of rpm is reached, it will remain constant at that value even though the acceleration applied is increased. This rpm value will then begin to decrease when acceleration is applied in the opposite direction, that is, when brakes are applied.

In case of automatic gear based driving, the rpm limits decide the gear shift occurrences. The rpm value can be considered as a representative of the speed of the vehicle. Whenever rpm corresponding to the maximum achievable torque value is reached, the gear is up shifted. This continues obviously, as long as the highest gear is not reached. Thereafter, the vehicle continues to drive in the highest available gear until speed is reduced. Similarly, when the rpm value falls to the set minimum limit, the gear is down shifted. Here again, this downshift will occur only so long as the first gear is not reached.

The starting gear ratio has been set to that corresponding to gear number 1 in case of automatic gear driven vehicles and has been set to neutral in case of manual gear driven vehicles. For the case of reverse gear, the gears ratio has been set to the ratio similar to gear number 1.

5. User Interface

User Programs allow the user to interact with the simulation using input devices like the steering wheel, a joystick or the mouse. In the AutoSim simulator, the 'Demo User Program' and the 'joystick User Program' provide a means for the user to interact with the simulation.

This section discusses one of my contributions to the project, the user program that provides an interface to both the joystick and the steering wheel, the steering wheel being the preferred interface as it is closest to reality.

5.1 Control Using Steering Wheel or Joystick

The user program 'joystick' can be used to control the vehicle with the steering wheel or the joystick.

The user is provided with a hardware interface that provides controls like the steering wheel to steer in the desired direction; brake and accelerator pedals to control acceleration and deceleration; forward and reverse gears to set the direction of motion of the vehicle; and various gears to enable better speed control. In addition to these, a gear mode control button is available to toggle between the automatic gear mode and the manual gear mode. Buttons are available for the users to turn on the various lights on the vehicle. These include head lights and the turning indicators.

Figure 22 shows the hardware available; along with the various controls assigned.

Figure 23 shows the joystick controls, which can be used as an alternative to the steering wheel. A joystick is good for testing purposes as it can be easily moved around as opposed to the steering wheel which remains fixed to a support.

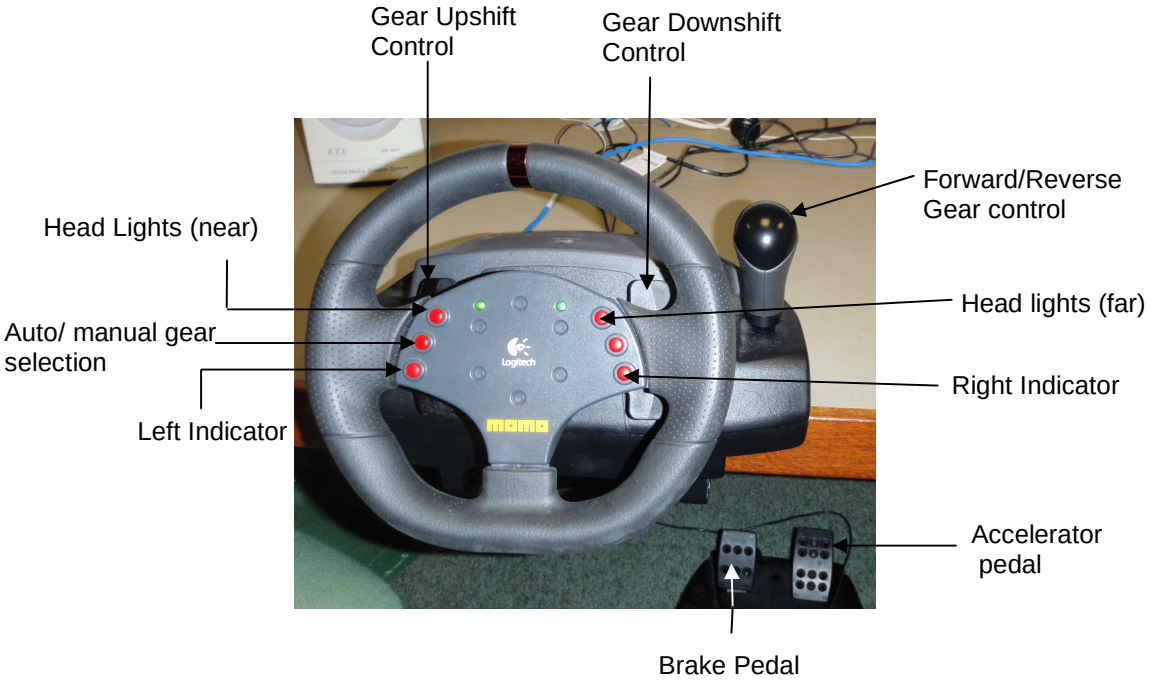


Figure 22 : Steering Wheel Controls

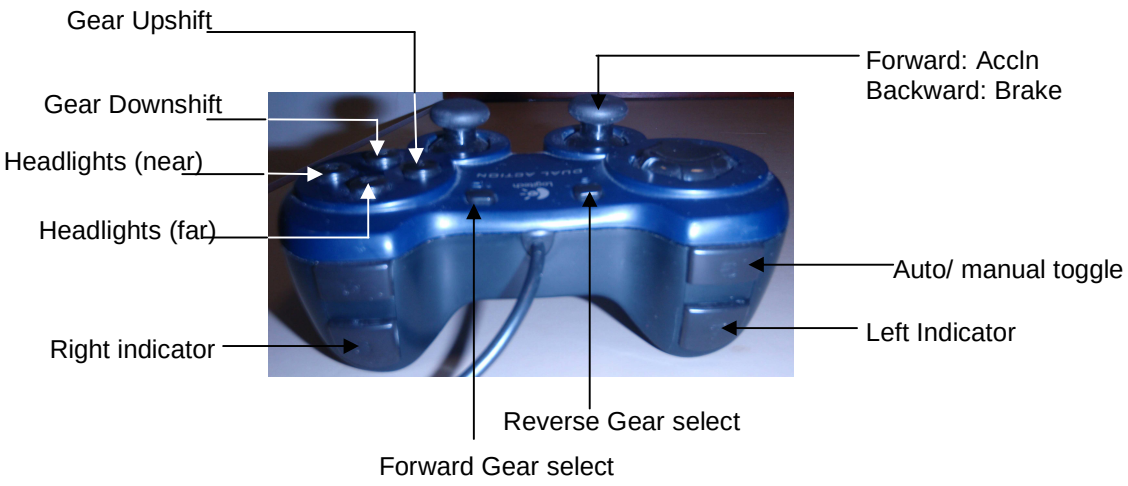


Figure 23: Joystick Controls

The user program sends the appropriate acceleration, brake and steering values to the drive actuator. When brakes are applied, the user program turns on the brake lights for the vehicle. Similarly, in case of indicator lights, when the turn is completed, the user program turns the corresponding indicator off. Also, it makes sure that the user or driver does not turn on both the right and the left indicators simultaneously.

This user program uses the velocity sensor. It has been used in the implementation of brakes. The braking force is nothing but a negative accelerating force or a decelerating force. The equation for longitudinal force, from section 4.2.1 clearly shows this. It is important that we stop applying the braking force as soon as the speed is reduced to zero otherwise the car will end up going in reverse. Similarly, in case the car is already traveling in the reverse gear, an excessive braking force will cause it to start moving forward.

Note that one of the buttons on the steering wheel is available for use in the future. An important point to remember while implementing it is the concept of 'key de-bounce' time. Since we are continuously polling the buttons to check for inputs from the user, it is important that the button press be considered valid only if it occurs after certain minimum time duration from the previous one (say 250 ms); otherwise a single button press will often be detected as multiple inputs resulting in unexpected and erroneous results.

6. Autonomous Driving

As specified previously, user programs control the robots in the simulation. Note that, the driven vehicle is not the only robot controlled by user programs. Other objects such as traffic lights or vehicles contributing to traffic on streets; that can change state are also specified as robots and controlled by user programs.

There are a few user programs that run in the simulation without requiring inputs from the driver. There is one that controls the changing of the traffic lights present at the intersections of streets. Then there is another that simply introduces a stationary vehicle with brakes on, into the simulation environment. The '*follow road*' user program uses image processing routines to enable a vehicle to drive autonomously along a given road. It is discussed in detail in [38].

The user program for autonomous driving, that was my contribution is discussed in detail in this section. Its aim is to generate moving traffic on the roads, so as to make the driving experience more realistic. This means creating a car with artificial intelligence (AI) that can drive on its own. Currently, it is written to drive a robot car around a block in a given world.

6.1 Sensors Used

The user programs for autonomous driving use the robot's virtual sensors to get data from the environment, make decisions depending on user inputs and some predefined criteria, and send appropriate control signals to the robot actuators. The autonomous robot has been instructed to prevent collisions by detecting obstacles, turn at the four intersections of the block, and drive straight on the road connecting the intersections. The various sensors used for achieving this functionality are the PSD sensors, the GPS sensor, the inclinometer sensor and the velocity sensor.

6.1.1 The PSD Sensor

The PSD sensors connected to the robot are used to detect and avoid potential collisions. Currently, the user program continuously monitors the reading from the

PSDs and applies brakes to the robot car if it detects any obstacle within a distance of 10 meters.

The *palPSDSensor()* function from PAL attaches a PSD sensor to a given body, at a specified position, and pointed in a given direction, with a specified maximum range. The parameters of position, direction and maximum range are specified in the xml file defining the robot. In my case, the PSD on the front side of the vehicle is being used.

The PSD sensor returns the distance between its location and direction and the nearest object. As per the PAL documentation, the process of doing so is assumed to involve casting a ray and determining the intersection point to calculate the distance between the objects. The PSD sensor can therefore be referred to as raycasting or raytracing. Figure 24 shows the body to which the sensor is attached (the cube in this case), the position at which the PSD sensor is located (the bottom of the arrow), and the direction which the PSD is facing (the tip of the arrow).

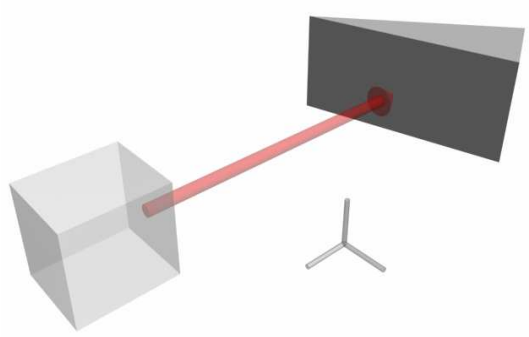


Figure 24: Ray casting in the PSD sensor. Figure from PAL Documentation [20]

This figure shows the operation of the PSD, showing the PSD ray intersecting a neighbouring object. The length of the arrow is the distance returned when the PSD is queried.

6.1.2 The GPS Sensor

The GPS sensor connected to the robot is used to decide when the robot has to turn at an intersection. Also, while driving on a straight road, the GPS sensor has been used to check for deviations from the straight line path. Both of these are

achieved by comparing the latitudes and longitudes and their corresponding differences from the desired values.

The GPS sensor returns the location of the center of the body as a GPS string. The *palGPSSensor()* function initializes the GPS sensor and attaches it to a given body. The string retuned by the GPS contains fields such as utc, latitude, longitude, velocity, date, magnetic variation etc. Currently, the only fields of interest are the latitude and the longitude values.

6.1.3 The Inclinometer Sensor

The Inclinometer sensor connected to the robot is used to decide when the robot has to stop turning at an intersection. The Inclinometer sensor returns the angle between its initial orientation and its current orientation.

The *palInclinometerSensor()* function initializes the Inclinometer sensor and attaches an inclinometer to a given body, with a given initial orientation. It returns the angle in radians, between the initial and the current orientation of the body, when measured in the plane normal to sensor axis. In this case, the initial orientation is along the Z axis, which is the axis along which the vehicle moves when travelling along a straight line. This can be set in the xml file defining the robot.

6.1.4 The Velocity Sensor

The velocity meter sensor connected to the robot is used to control the robot speed. It gives the velocity in a given direction.

The *palVelocimeterSensor()* function initializes the velocity sensor and attaches an it to a given body, with a given heading. It returns the linear velocity of the body. The heading is along the Z axis, which is the axis along which the vehicle moves when travelling along a straight line. This too can be set in the xml file defining the robot.

6.2 Autonomous Driving Control Mechanisms

As discussed in the previous section, the GPS output determines the point of turning and the inclinometer sensor helps to keep track of the turning angle. Between

the turns, the robot has to drive along a straight line path. Hence, the need for some sort of steering control arises. This section considers the various types of control mechanisms possible.

6.2.1 On Off Control

An on off controller is a two state controller. In many applications, the controller will be either on or off. However, to achieve optimal control, the controller is designed to have two extreme values. If the input value is above a given threshold, the controller will select one of the extremes as the output and if the input is below the threshold, the other extreme will be selected as the output. Since, this controller restricts every control signal to take on one of the two limiting values; it is also referred to as a bang-bang control.

In case of autonomous steering wheel control, the input to the controller is the deviation of the robot from the desired position. Depending on the direction of deviation, the vehicle steering value can be selected as either positive or negative. That is, if the robot has moved to the left of the desired path, a positive steering value will turn it towards the right and similarly, a negative steering value can be used to turn a robot left when it deviates to the right. The problem with this type of controller is that the steering control value will always be switching between the two bounds irrespective of the amount of deviation; and hence the vehicle will never travel along a straight line path.

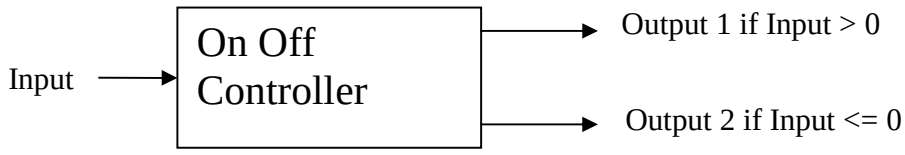


Figure 25: On Off Controller

6.2.2 Proportional Control

A proportional controller is a feedback controller that takes into account the amount of output deviation from the desired value. It generates an output proportional to the error signal, where the error signal is the difference between the actual signal and the available signal. It can be expressed in terms of the equation from [64]:

$$P_{\text{out}} = K_p\, e(t)$$

Where,
P_{out} = Output of the proportional controller
K_p = Proportional gain
e(t) = error signal at time t

In case of steering wheel control, the error signal is a measure of the deviation of the robot from the desired straight line position. The gain needs to be selected carefully as a small gain causes a slow response and a very high gain can cause instability or oscillations. The Proportional Controller will always have a residual error which is a function of the gain selected.

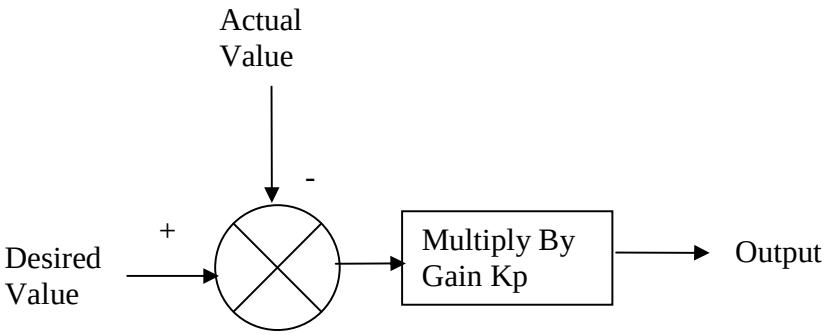


Figure 26: Proportional Controller

6.2.3 PID Control

The PID controller [65] can be seen as a combination of three separate controllers; namely the Proportional, the Integral and the Derivative controller. The Proportional controller, as described in the previous section, has an output proportional to the existing error and it works towards reducing the overall error. The output of the Integral controller is dependent on the sum of errors, and it works towards accelerating the error reduction process. Summing the error over time produces a signal large enough to eliminate the residual error that occurs with the proportional controller. The Derivative controller does not play any role in reducing the final error. Its output is dependent on the rate of error change. It works towards improving the overall stability of the system by reducing the error change rate. It minimizes overshoot and ringing effects, which may be caused by the integral controller when responding to accumulated errors.

The output of the PID controller is a weighted sum of the outputs of these three controllers. It can be given as [65]:

$$\text{Output} = P_{out} + I_{out} + D_{out}$$

Where,
P_{out} = output of the Proportional Controller
I_{out} = output of the Integral Controller
D_{out} = output of the Derivative Controller

The equation for the output of the Proportional Controller is explained in the previous section and the other two terms can be expressed as follows (from [65]):

The Output of the derivative controller is given by:

$$D_{out} = K_d \frac{de}{dt}$$

Where,
K_d = Derivative gain
e = error signal
t = current time.

The Output of the Integral controller I_{out} is given as:

$$= K_i \int_0^t e(\tau)\, d\tau$$

Where,
K_i = Derivative gain
e = error signal
τ = past time.

The figure 27 below shows the block diagram of a PID controller.

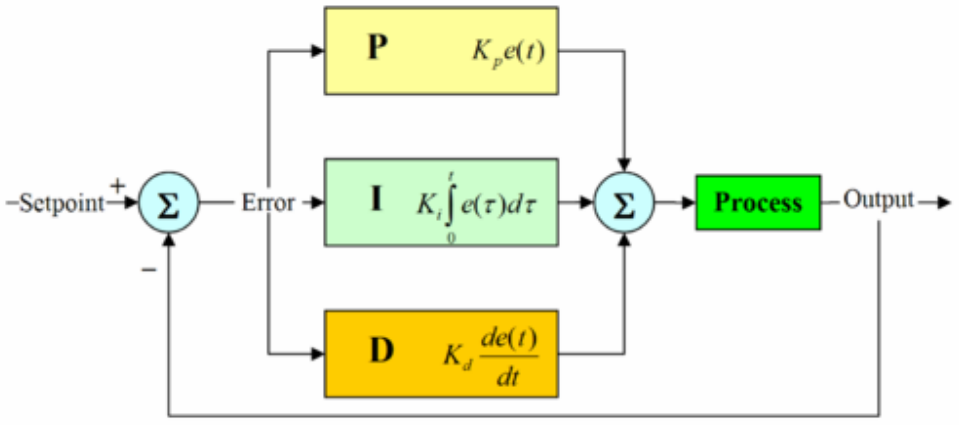


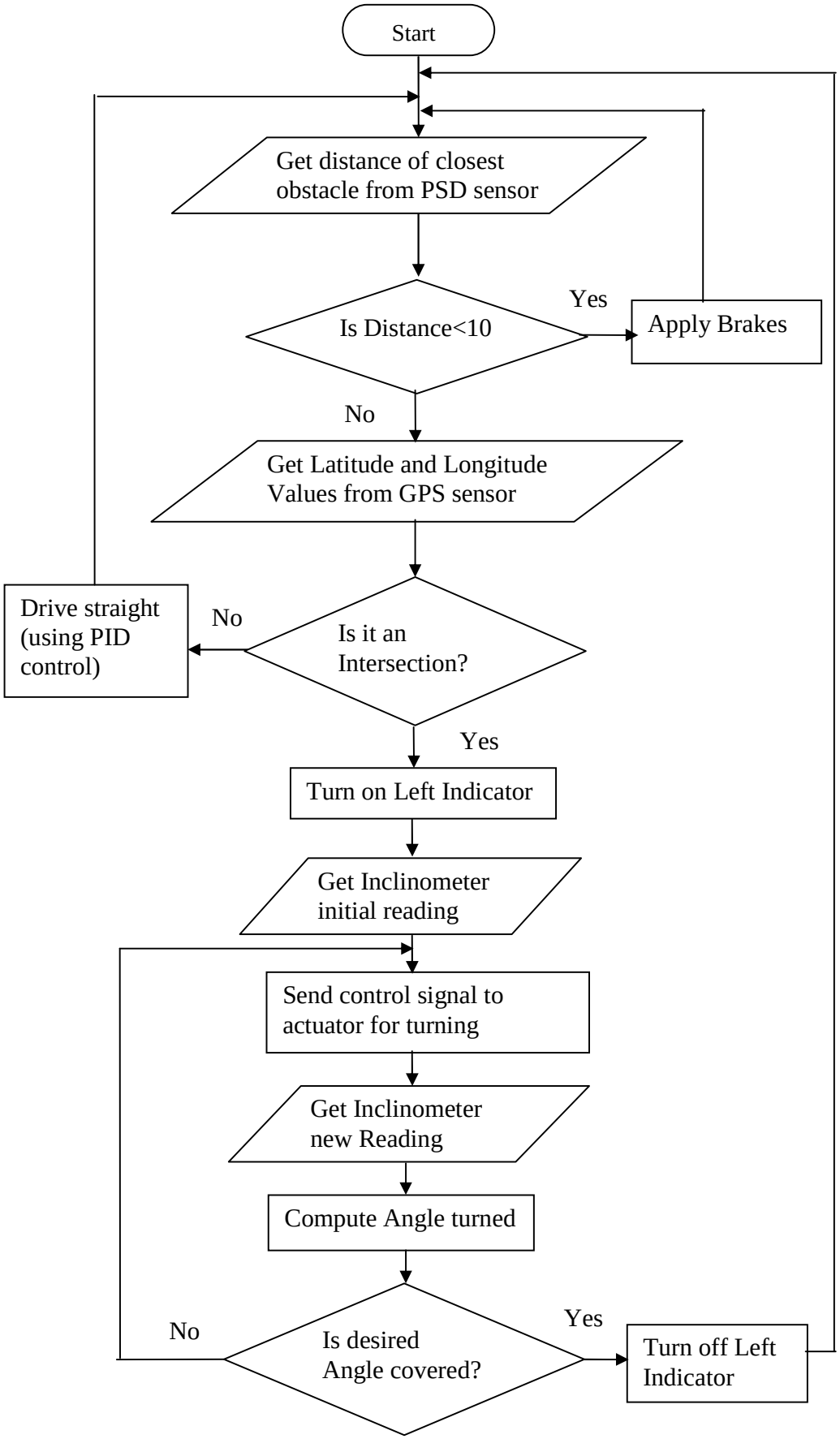
Figure 27: PID Controller. Figure from [65]

Proper selection of the gain parameters for the three controllers determines the stability and effectiveness of the control process. This selection is referred to as tuning the controller. My implementation of the PID controller in the autonomous driving user program makes use of similar work done in the past, as part of the AutoSim project, for the implementation of autonomous driving routines using image processing. The thesis [38] explains the details of the tuning procedures followed.

6.3 Driving Algorithm

The driving algorithm is designed to turn the autonomous robot at the four intersections of a block and drive straight between the intersections. While doing so, it is instructed to apply brakes if any obstacle is detected. Also, brake and indicator lights should be turned on and off at the proper times.

The flowchart below shows the steps followed by the algorithm.


Start
Get distance of closest obstacle from PSD sensor
Is Distance<10
Yes
Apply Brakes
No
Get Latitude and Longitude Values from GPS sensor
Drive straight (using PID control)
No
Is it an Intersection?
Yes
Turn on Left Indicator
Get Inclinometer initial reading
Send control signal to actuator for turning
Get Inclinometer new Reading
Compute Angle turned
No
Is desired Angle covered?
Yes
Turn off Left Indicator

7. Test Scenarios

This section discusses the various experiments performed to test the implemented functionality. The experiments involve testing the gear functionality, the wind gust simulations and the autonomous driving. For conducting all tests, simple world files, with appropriate markings were created.

7.1 Gear Functionality Testing

Tests were performed to check the change in vehicle behavior due to implementation of gears. The tests conducted and the results obtained are summarized below.

7.1.1 Tests Performed

The following tests were performed to test the gear implementation:

1. Velocity at which the gear shifts occur in automatic gear based cars was noted. The gear shifts occur when the RPM values corresponding to the maximum torque produced by the engine are reached. This maximum value can be seen from the torque - rpm curve in figure 21 Note that this curve and the corresponding equation can be different for different cars.

2. Time to travel 1000m for automatic and manual cars was noted in identical environments and compared. The same car was used in both cases, with a changed mode of gear control.

.

3. In case of a manual gear based car, the time to reach 30 km/hr in different gears was recorded.

Separate test conditions were simulated for carrying out the above experiments. A simple world was constructed consisting of a clear, straight road without any gradient. No houses, trees or traffic lights were included. However, markings were placed at every 100m so as to enable distance measurement. The road was free from any other traffic so as to ensure an obstacle free environment. Also in all the cases, the accelerator pedal fully pressed so as to ensure uniformity of input. Figure 28 shows a screen shot of the test environment.

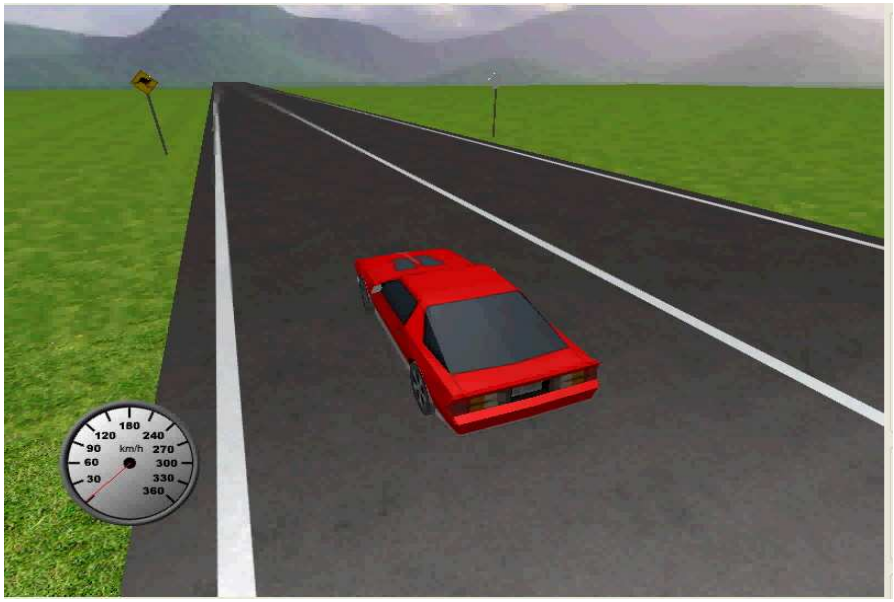


Figure 28: Obstacle free test environment with sign board markings

An alternative test environment is one consisting of a road only. In this case white lines are marked on the road at every 100m. Figure 29 shows a screen shot of this test environment.

Figure 29: Alternative test environment with line markings on the road

7.1.2 Test Results

1. Velocity at which the gear shifts occur in automatic gear based cars

In automatic gear based cars, gear shifts occur when the RPM values corresponding to the maximum torque produced by the engine are reached. This maximum value depends on the torque - rpm curve as specified before. To illustrate the importance of the torque - rpm curve, this test was performed for two curves. Below are the curves and the corresponding results.

Curve 1:

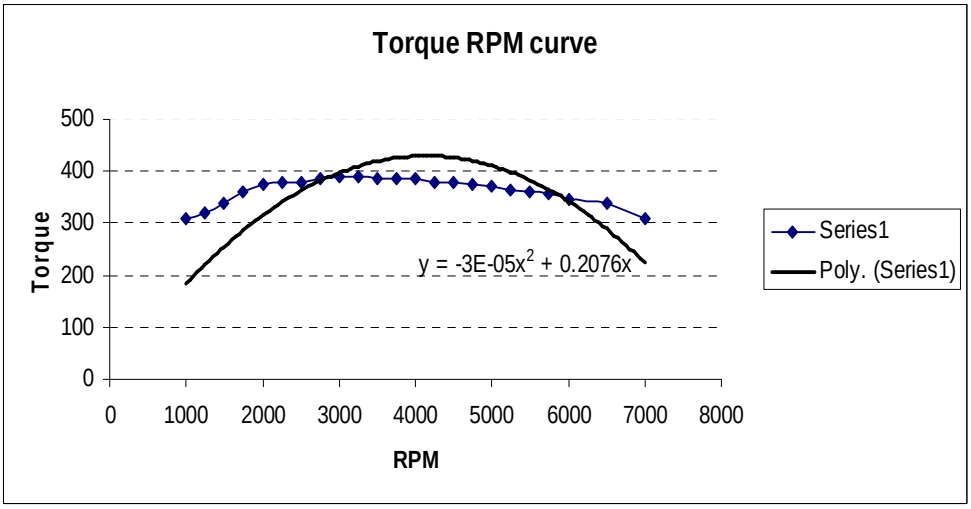


Figure 30: Torque RPM curve 1

Up shifts:

From Gear	To Gear	Vehicle Speed (km/h)
1	2	28
2	3	45
3	4	62
4	5	105

Table 1: Gear Upshift Speeds in an automatic gear vehicle

Down shifts:

From Gear	To Gear	Vehicle Speed (km/h)
5	4	64
4	3	48
3	2	27
2	1	20

Table 2: Gear down shift Speeds in an automatic gear vehicle

The Figure 30 shows the curve and the tables 1 and 2 give the gear shift speeds for the curve selected for use in AutoSim.

Curve 2:

The Figure 31 shows another curve used for testing; and the tables 3 and 4 give the gear shift speeds for this curve.

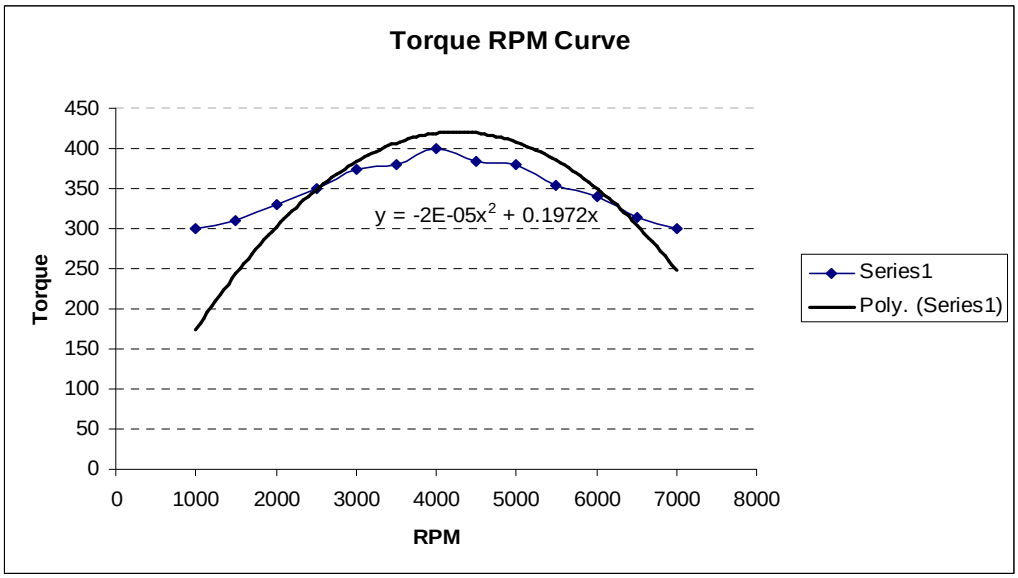


Figure 31: Torque RPM curve 2

Up shifts:

From Gear	To Gear	Vehicle Speed (km/h)
1	2	40
2	3	60
3	4	82
4	5	150

Table 3: Gear Upshift Speeds in an automatic gear vehicle

Down shifts:

From Gear	To Gear	Vehicle Speed (km/h)
5	4	78
4	3	63
3	2	39
2	1	20

Table 4: Gear down shift Speeds in an automatic gear vehicle

2. Time to travel 1000m for automatic and manual cars

Under identical test conditions, the time to travel 1000m for both automatic and manual gear based cars was noted. Note that for this test, only the curve 1 is used, since that was selected for implementation in AutoSim.

In the first case, the gear control mode of the car was set to automatic and the time required for it to travel a span of 1000m was noted.

In the second case, gear control mode of the car was set to manual and the time required for it to travel a span of 1000m was noted, allowing the driver to shift gears as per his or her own free will. However, in this case, the time value will vary from driver to driver since this time is related with the time or the speed at which the driver decides to shift gears. Hence, the value specified below was taken as an average of 3 timings. Lastly, the time taken to travel the distance completely in first gear was noted.

Time to travel 1000m

For an automatic gear driven car: 78 sec

For a manual gear driven car using 5 gears: 91 sec

For a manual gear driven car using 1st gear only: 146 sec

3. Time to get a car to 30 km/hr using different gears

Ideally, a driver should start moving a manual gear driven car in the first gear, to be able to reach the desired rpm soon. Here, the car is started in the first gear and as soon as it reaches a speed of 1 km/h (considered as set into motion), the gear is switched. Then the times taken to get a car to a speed of 30 km/h are noted in cases of using different gears. This test again was implemented for the selected curve 1 alone.

Table 5 below shows the values observed.

Gear Number	Time (seconds)
1	6
2	11
3	23
4	Does not reach 30 km/hr
5	Does not reach 30 km/hr

Table 5: Time to get a car to a speed of 30 km/hr using different gears

Study of Effect of gear change on velocity and RPM

In order to study the engine behavior in the presence of gears, a graph was plotted showing change in velocity with time and change in RPM with time; along with the impact of a gear shift. Figure 32 shows the change in velocity and the change in rpm with time and the effect of a gear change.

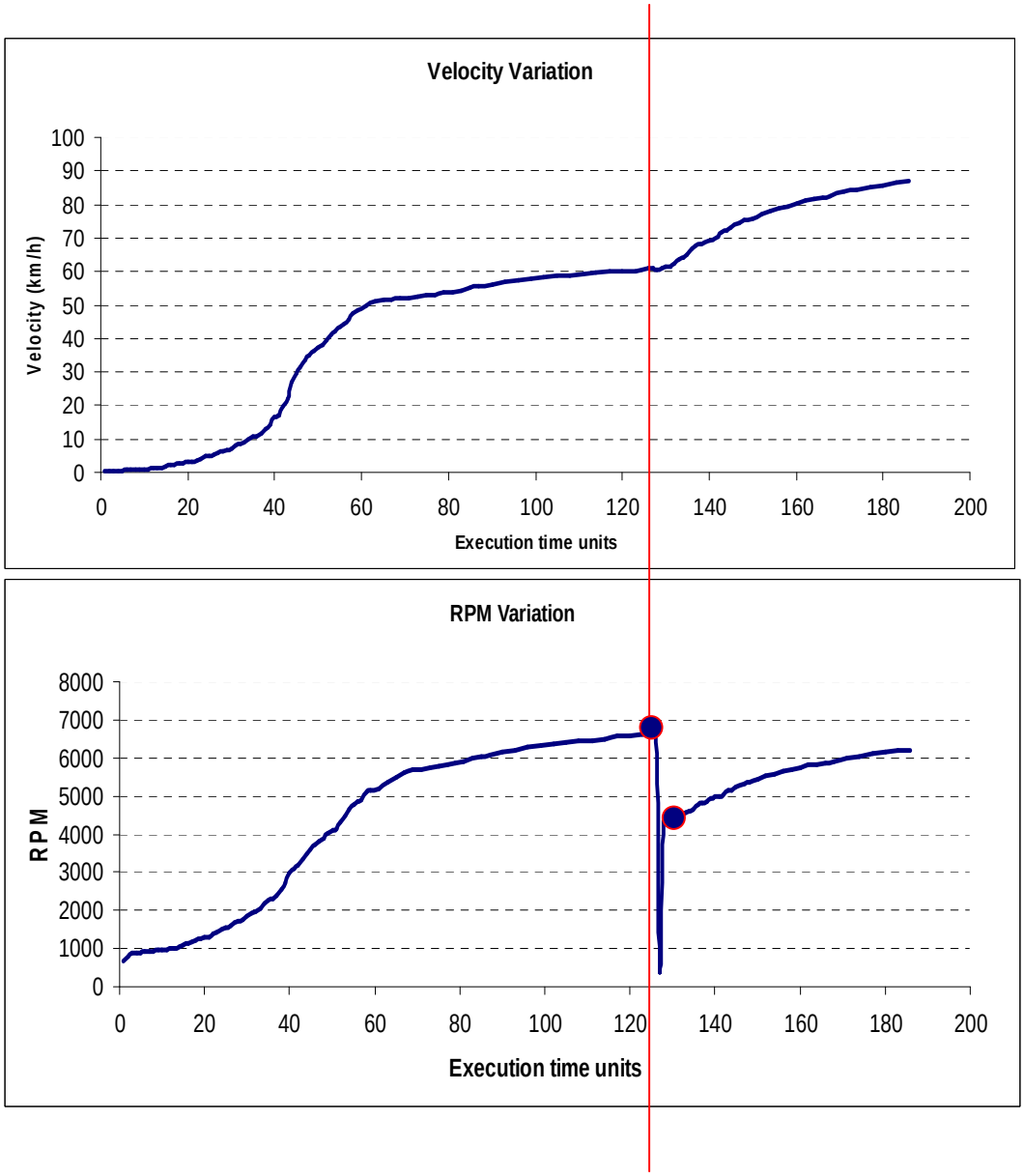


Figure 32: Effect of gear shift on velocity and RPM variations

7.1.3 Conclusions and Discussions from the Results

From the first test it can be seen that the velocity at which gear shifts occur will vary in different vehicles, depending upon the engine and it is possible to simulate this variation by replacing a single equation. Also, the results obtained for gear shifts are reasonably good, that is, as expected especially with the first curve.

The second test shows that for a given distance, the time required by an automatic gear controlled vehicle is less than that required by a manual gear controlled vehicle. This will be true in most cases as the drivers are not likely to change gear at the exact moment when the engine torque reaches its maximum value. Also, the case of driving the entire distance in the first gear shows the importance of gears and represents the consequence of amateur drivers not changing gears in time. Overall, the value provided by automatic gears can be observed and the performance verified as matching with the expected.

Test three basically serves to prove that the implementation behaves in a realistic manner. When a higher gear is selected at a low speed, it takes higher amount of acceleration and a longer time for the vehicle to reach a given speed.

From the velocity variation curve, it can be seen that the velocity increases at a faster rate initially, and then it increases at a slow pace. The corresponding RPM variation graph shows that the peak rpm that the engine can produce has reached and it is necessary to change the gear to the next higher one. The vertical line shows the point of gear change. It is clear that the velocity now again begins to rise at a fast rate. The RPM curve drops down momentarily showing the simulation of the neutral gear. It then rises to an RPM value corresponding to the velocity before the gear change, but with the new gear ratio. The two large dots show the RPM values just before and after the gear shift.

7.2 Testing Wind Gust Simulations

For testing the wind gust simulation, the slider bars are moved to the extreme positions to make it easy to notice the effect. It can be observed that when a wind of a speed of about 250 km/hr blows along the direction of motion of the vehicle, a stationary vehicle will slowly begin to move forwards. Similarly, a wind of a speed of about 250 km/hr against the direction of motion of the vehicle will cause the

vehicle to slowly move backwards. When the strong wind is blowing in a sideways direction, the vehicle can be seen shaking.

The effect on a moving vehicle has been observed by noting the amount of acceleration required to attain a certain speed in the presence of a strong wind in the opposing direction, a strong wind in the direction of motion and in the absence of abnormal wind conditions. Similarly, the amount of braking force required in these three conditions is noted. The results of these tests are summarized below:

Time required to attain a speed of 30 km/ hr at a given acceleration:

The time required to attain a speed of 30 km/h when the accelerator pedal is fully pressed, is noted for different wind conditions.

Wind Condition	**Time (sec)**
Normal	7.5
Forward wind of 250 km/h	6.1
Opposing wind of 250 km/h	11.0

Table 6: Effect of wind force on acceleration

Time required for braking from a speed of 30 km/ hr at a given braking force:

The time required to reach a to a stop (that is a speed of 0 km/hr) from a speed of 30 km/h when the brake pedal is fully pressed, is noted for different wind conditions.

Wind Condition	**Time (sec)**
Normal	2.7
Forward wind of 250km/h	4
Opposing wind of 250 km/h	2

Table 7: Effect of wind force on braking

It can be easily seen from the values that wind force in the direction of motion helps motion and wind force against the direction of motion retards it; thus affecting the time required to attain the same speed with the same amount of

acceleration. The reverse is true in case of braking. A wind force in the direction of motion opposes the braking force while a wind force against the direction of motion helps the braking.

7.3 Testing Autonomous Driving

A test scenario was set up to test the driving around the block. The environment created was free from obstacles and did not include any unnecessary objects like houses or trees. The objective was to ensure that the autonomous vehicle will turn at the specified four intersections and travel along a straight line between them.

The angle of turning at each intersection needed to be set by experiment as the roads did not intersect at an exact angle of 90˚.

The important test was the one for steering control to ensure driving along a straight line path. As discussed in section 6.2, a number of control mechanisms are possible. Each of these was implemented with the aim of comparing their performances.

Figure 33 shows the path taken by the robot when using different controllers while traveling along a straight road. The x and y axes relate to the variation in latitude and longitude along the robot's path. Note that the complete path will be rectangular in nature. The figure considers only one section of straight road between two turns. It is assumed that driving along all four sections will yield similar results. For driving along a straight line through the section considered here, the latitude will vary and the longitude must remain constant. Any change in longitude indicates a deviation from the ideal path.

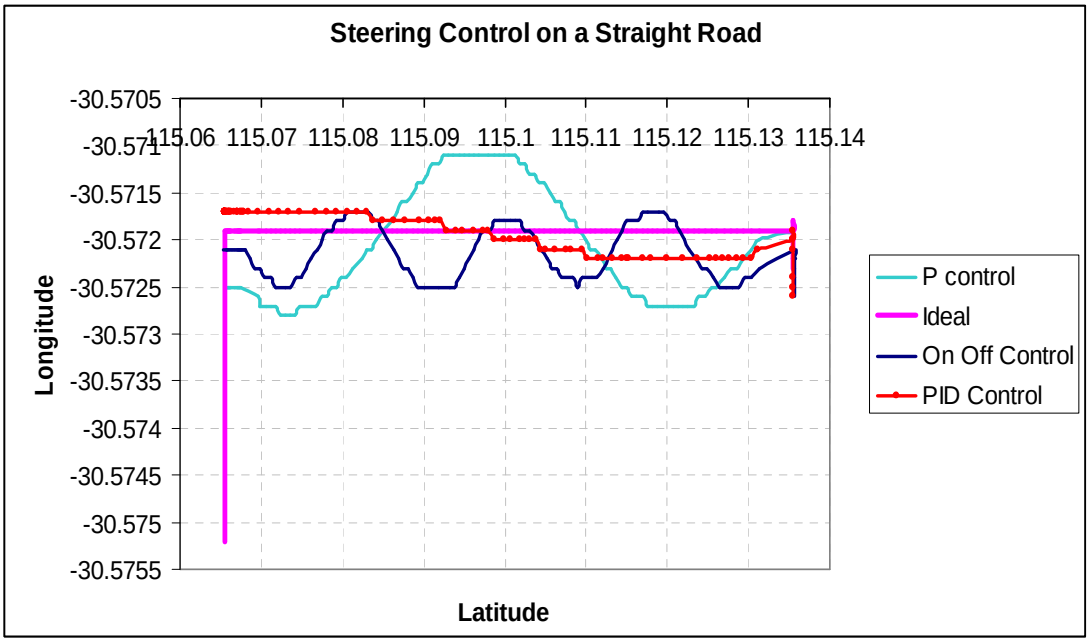


Figure 33: Controller Performance Comparison plot

By comparing the path traveled using each of the three implemented controllers with the ideal path; the following observations can be made:

- As expected, the on off controller shows oscillatory behavior and hence it is not suitable for use in steering control.
- The Proportional (P) controller has reduced oscillations as compared to the on off controller, however the oscillations still exist. Note that the magnitude of path variation introduced by use of the P controller can be reduced by further tuning the proportional gain.
- It is clear that the PID controller is the closest in approximating the straight line path.

Hence, the PID controller has been selected for final implementation of autonomous driving.

Another test scenario was set up to confirm obstacle detection. Here a vehicle on brakes, acting as an obstacle; was placed at some distance in front of the autonomous vehicle. The objective was to ensure that the autonomous vehicle will not collide with it. When the simulation started, the robot car drove straight until it reached close to the obstacle and when it was within 10m from the obstacle, it applied brakes and came to a halt.

8. Conclusions

The most important aspect of an automotive simulator is realism. The physics of a driving simulator plays an important role in achieving the resemblance to reality. The use of an already existing physics engine, Bullet, along with the Physics Abstraction Layer (PAL) greatly simplified this task. However, the implementation of gears, which has not been included in PAL, was necessary to simulate real world behavior. The velocity graph of figure 32 clearly shows the effect of introducing gear physics. In absence of gears, this graph would show a linear increase in velocity.

Another important aspect of a driving simulator is providing test conditions which are difficult to generate and dangerous to test in real life. This includes extreme weather conditions. The first step towards this, that is, the simulation of strong winds enables us to study the effect of an increased wind force on vehicle physics. Other conditions that may be simulated are discussed in the next chapter.

In addition to realistic vehicle physics, adding realism to the driving environment is important; especially in terms of having moving traffic and stationary obstacles on roads. Autonomous driving around a block goes a step towards this. In addition, it demonstrates the use of sensors in making vehicle behavior related decisions, which can be used for incorporating adaptive cruise control features into user driven vehicles.

Thus with increased realism, AutoSim can be used for driver training; for testing complex driver assistance functions being introduced by car manufacturers; for testing vehicle behavior in dangerous and difficult to control situations and for computing performance variables that are hard to measure experimentally.

9. Future Work

A lot more work needs to be done to bring the AutoSim simulator close to simulating realistic conditions.

Simulating the effects of weather is an important feature of a simulator. Simulating wind gusts can be considered as the first step in this direction. Further from this, other weather conditions such as snow or rain can be simulated. This includes both the simulation of the change in clarity of drivers' view in the graphics and the simulation of the change in the friction coefficient of the wheels with the road, in the physics. With respect to the change in clarity of drivers' view in the graphics, other environmental conditions such as lighting conditions and dust behavior can also be simulated. Various sounds can also be added to increase realism.

The autonomous driving user program is a step towards creating moving traffic; however it currently simulates a single vehicle going around a block. In the future, AutoSim needs to simulate more traffic, and introduce new robots like cyclists and pedestrians. This will make the simulation world look more realistic. Moreover, currently the autonomous robot has no interaction with traffic lights. So, the autonomous robots need to obtain information about the state of the traffic signals and modify their actions accordingly so as to avoid passing through a red light. Also, in the present case, if the autonomous robot happens to reach behind a parked car or any stationary object, the PSD sensors will detect it as an obstacle and apply brakes to avoid collision. However, since the object is stationary, our autonomous car will permanently remain in that position with brakes on. Hence, routines need to be included to detect such situations and enable the autonomous robot to overtake stationary vehicles or other objects.

The user program using joystick or the steering wheel currently does not provide for varying steering ratios for different cars. This is because; the car xml does not specify this. In the future, an additional tag can be added in the robot xml file to include the steering ratio applicable to the particular car model and the corresponding value be considered in the driving routines. This will help understand the difference between cars. Also, some features developed for autonomous driving can be implemented in the user controlled driving as a step towards adaptive cruise control. For example, along the lines of the obstacle detection functionality from the autonomous driving routines, the PSD sensors from cars can be used to maintain a

fixed distance from the vehicle in front. Similarly the velocity meter sensor can be used to set speed limitations on the vehicle.

Another important development necessary is a means to provide graphical displays of physics parameters. For example, a dynamometer on the dashboard will help visualize the effect of change in gears. It will facilitate the study of change in rpm with change in gear; and the relationship between velocity, gear number and engine revolutions can be understood more easily. Also, in the simulation of wind effects, it will be advantageous if the user has a choice to display arrows along the direction of the wind flow. This will make it easier to visualize the effects of strong gusts of wind and will be helpful specially during testing.

Other than these, some means to improve AutoSim are by adding more vehicle models, more driving environments and different traffic scenarios. It is a good idea to provide an environment with UWA and its surrounding area. Maps of some major cities can also be used for building realistic worlds. Currently, work is being done on making AutoSim cross platform. AutoSim has been implemented using a client-server architecture. The aim of this is to enable the running of multiple clients simultaneously. This will be one of the goals in the future.

10. Abbreviations

AC: Alternating current

ACC: Adaptive Cruise Control

AI: Artificial Intelligence

API: Application Programmer Interface

CIIPS: Centre for Intelligent Information Processing Systems

COLLADA: COLLAborative Design Activity

DARPA: Defense Advanced Research Projects Agency

DC: Direct current

DOF: Degree of freedom

DOM: Document Object Model

DTD: Document Type Definition

GPS: Global Position Sensor

GUI: Graphical User Interface

IBDS: Impulse Based Dynamics Simulation

INS: Inertial Navigation System

LAN: Local Area Network

ODE: Open Dynamics Engine

OSM: Open Street Map

PAL: Physics Abstraction Layer

PID: Proportional Integral Derivative

PSD: Position Sensitive Device

RARS: Robot Auto Racing Simulator

RPM: Revolutions Per Minute

SDL: Simple Direct Media Layer

TORCS: The Open Race Car Simulator

UTC: Universal Time Coordinates

UWA: University Of Western Australia

XML: eXtensible Markup Language

11. References

[1] ‘DARPA Grand Challenge’;
http://en.wikipedia.org/wiki/DARPA_Grand_Challenge; February 2008.

[2] ‘Car Sim’; http://www.carsim.com/applications/; February 2008.

[3] ‘Reactive Navigation of Autonomous Guided Vehicle Using Fuzzy Logic’ Ir. Idris Ismail and Mohd. Fariz Nordin, 2002 Student Conference on Research and Development Proceedings, IEEE.

[4] ‘Onboard Real-time System for 3D Urban Environment Reconstruction’, I. ABUHADROUS, F. NASHASHIBI, C. LAURGEAU, F. GOULETTE, Center of Robotics Ecole des Mmes de Paris, IEEE 2003.

[5] ‘Current Status and Future Plans for Digital Map Databases in Japan’, Masao Shibata, Yasuomi Fujita, Japan Digital Road Map Association, IEEE1993.

[6] ‘Producing a Three-dimensional Road Map for Car Navigation’ , Kouhei Toul, Tohru Irie2, Joo Kooi Tan', Seiji Ishikawal, Department of Mechanical and Control Engineering, Kyushu Institute of Technology, Fukuoka, Japan, SICE-ICASE International Joint Conference 2006.

[7] ‘Status and Directions of Digital Map Databases in Europe’, Hinrich Claussen, Robert Bosch GmbH, IEEE - IEE Vehicle Navigation & Information Systems Conference, Ottawa - VNlS '93

[8] ‘Adaptive Fuzzy-Network-Based C-Measure Map-Matching Algorithm for Car Navigation System’, Sinn Kim and Jong-Hwan Kim, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 48, NO. 2, APRIL 2001

[9] ‘Fusion of Sensor Data in Siemens Car Navigation System’, Dragan Obradovic, Member, IEEE, Henning Lenz, and Markus Schupfner, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 56, NO. 1, JANUARY 2007.

[10] ‘Data analysis for parallel car-crash simulation results and model optimization’, Liquan Mei, C.A. Thole, Science Direct, Volume 16, Issue 3, March 2008, Pages 329-337.

[11] 'Development of Universal Driving Simulator with Interactive Traffic Environment' Yoshihiro Suda, Yoshiyuki Takahashi, Masao Kuwahara, Shinji Tanaka, Katsushi Ikeuchi, Masataka Kagesawa, Tomoyoshi Shraishi, Masaaki Onuki, Ken Honda and Makoto Kano, IEEE 2005.

[12] 'A Low-Cost Driving Simulator for Full Vehicle Dynamics Simulation', Andy R. W. Huang and Chihsiuh Chen, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 52, NO. 1, JANUARY 2003.

[13] 'SDL - Simple Direct Media Layer', http://www.libsdl.org/; March 2008.

[14] 'Modeling of Virtual Traffic Environment For Driving Simulator', Yang Nanyue, He Hanwu, Lou Yan, Lu Yongming, IEEE, 2006.

[15] OpenStreetMap, http://www.openstreetmap.org/; February 2008.

[16] 'The Bullet Physics Engine'; http://www.bulletphysics.com/Bullet/wordpress/bullet; May 2008.

[17] 'TinyXML'; http://en.wikipedia.org/wiki/TinyXml; May 2008

[18] 'RakNet'; http://linux.softpedia.com/get/Programming/Libraries/RakNet-19718.shtml; April 2008.

[19] 'IRRLICHT Features'; http://irrlicht.sourceforge.net/features.html; April 2008.

[20] 'Physics Abstraction Layer'; http://www.adrianboeing.com/pal/index.html; March 2008.

[21] 'Probabilistic assessment of road vehicle safety in windy environments', J.Th. Snæbjornsson, R. Sigbjornsson, Engineering Research Institute, University of Iceland, Hjardarhaga 2-6, 107 Reykjavı´k, Iceland; C.J. Baker, School of Engineering, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK; Journal of Wind Engineering, and Industrial Aerodynamics 95 (2007) .

[22] 'Wind forces', Graham R Greatrix, *Accident Analysis Consultants.*

[23]' Effects of Non-Steady Aerodynamic Forces on Car Handling: Implementation in ADAMS of a New Experimental-Numerical Model for Longitudinal and Lateral Wind', F. Cheli, P. Dellachà, P. Mandelli, A. Zasso Politecnico di Milano, Milano, Italia.

[24] 'AERODYNAMICS OF ROAD VEHICLES', Wolf-Heinrich Hucho Ostring, Germany; Gino Sovran, General Motors Research and Environmental Staff, Michigan; Annual Reviews; www.annualreviews.org/aronline.
[25] 'Aerodynamics'; http://www.citroenet.org.uk/miscellaneous/aero/aero01.html; August 2008.

[26] 'Aerodynamics of Race Cars', Joseph Katz, Department of Aerospace Engineering, San Diego State University, San Diego, California; The Annual Review of Fluid Mechanics, Copyright c_ 2006.

[27] 'Effects of wind barrier on a vehicle passing in the wake of a bridge tower in cross wind and its response', S. Charuvisit, K. Kimura, Y. Fujino; Journal of Wind Engineering and Industrial Aerodynamics 92.

[28] 'Real-Time Simulation of Dust Behavior Generated by a Fast Traveling Vehicle'; JIM X. CHEN, XIAODONG FU, and EDWARD J. WEGMAN; Computer Graphics Laboratory, George Mason University.

[29] 'Gear Ratios'; Wikipedea; http://www.answers.com/topic/gear-ratio; August 2008.

[30] 'Elation electric bikes – Gear Theory';
http://www.elationebikes.com.au/gear_theory.htm

[31] 'Theory of Ground Vehicles'; J. Y. Wong, www.books.google.com

[32] 'How Automatic Transmissions Work'**;** Karim Nice;
http://auto.howstuffworks.com/automatic-transmission.htm; August 2008.

[33] 'Tech Center: Manual Transmission Basics'**;** Miles Cook;
http://www.edmunds.com/ownership/techcenter/articles/46029/article.html;
September 2008.

[34] 'Research on Fuzzy Logical Control System of the Electron Automatic Transmission of Automobile', Tu QiaoLing, Wan PeiLin; Chongqing institute of technology department of science & research; CHINA.

[35] 'The Numbers Game - Gear Ratio Calculating'; By Bob Mehlhoff; July 2008;
http://www.chevyhiperformance.com/techarticles/148_0208_gear_ratio_calculating/index.html

[36] 'Physics for a 3D Driving Simulator', Torsten Sommer, Executed at Robotics and Automation Lab, Centre for Intelligent Information Processing Systems, University of Western Australia, Perth.

[37] 'Graphics for a 3D Driving Simulator'; Johannes Georg Brand; Executed at Robotics and Automation Lab, Centre for Intelligent Information Processing Systems, University of Western Australia, Perth.

[38] 'Evaluation of a Vision based Driver Assistance System in Simulation'; Pål Simen Ruud; Centre for Intelligent Information Processing Systems (CIIPS); University of Western Australia; June 2008.

[39] 'Qt Cross-Platform Application Framework'; http://trolltech.com/products/; September 2008.

[40] 'Car Physics for Games'; Marco Monster; http://home.planet.nl/~monstrous, July 2008.

[41] 'The Physics of Racing'; Brian Beckman; Reformatted 25 April 08; http://phors.locost7.info/contents.htm; August 2008.

[42] 'Vehicle Dynamics for Racing Games'; Ted Zuvich; www.gamasutra.com/features/gdcarchive/2000/zuvich.doc; August 2008.

[43] 'Car physics – reference'; http://www.racer.nl/reference/carphys.htm; August 2008.

[44] 'Hans B. Pacejka'; Wikipedea; http://en.wikipedia.org/wiki/Pacejka; August 2008.

[45] 'Vehicle Design and the Physics of Traffic Safety'; Marc Ross, Deena Patel and Tom Wenzel; Physics Today; American Institute of Physics; January 2006.

[46] 'Physics & Math : Roll Over, Newton'; Curtis Rist; published online April 1, 2001; http://discovermagazine.com/2001/apr/featnewton; October 2008.

[47] 'The Physics of Sports Utility Vehicle Rollover Accidents'; C Johnson, Physicist, Physics Degree from Univ of Chicago; http://www.mb-soft.com/public/rollover.html; July 2008.

[48] 'Modeling Vehicle Rollover'; *Jonathan Hall and Jerry Magraw;* Penn State Erie, Th e Behrend College, Erie, PA; 'The Physics Teacher'; Volume 43; 2005.

[49] 'Rollover of Sport Utility Vehicles'; *Desmond N. Penny;* Southern Utah University, Cedar City, UT; THE PHYSICS TEACHER, Vol. 42, February 2004

[50] 'SubSim User's Manual'; http://robotics.ee.uwa.edu.au/auv/subsim/doc/About.html; October 2008.

[51] 'EyeSim - EyeBot Simulator'; http://robotics.ee.uwa.edu.au/eyebot/doc/sim/sim.html; October 2008.

[52] 'Sequential manual transmission '; http://en.wikipedia.org/wiki/Sequential_manual_transmission; October 2008.

[53] 'A scale relating tropical cyclone wind speed to potential damage for the tropical pacific ocean region: a user's manual'; Charles 'Chip' Guard, Mark A. Lander.

[54] 'Honda Introduces All-New Automobile Driving Simulator'; http://world.honda.com/news/2001/c010417_2.html; October 2008.

[55] 'Toyota Develops World-class Driving Simulator'; http://www.toyota.co.jp/en/news/07/1126_1.html; October 2008.

[56] 'TORCS, The Open Racing Car Simulator'; http://torcs.sourceforge.net/; February 2008.

[57] 'GRacer - 3D Motor Sports Simulator'; http://gracer.sourceforge.net/; February 2008.

[58] 'Robot Auto Racing Simulator'; http://rars.sourceforge.net/; February 2008.

[59] 'Vamos Automotive Simulator'; http://vamos.sourceforge.net/; February 2008.

[60] 'VDrift'; http://vdrift.net/; October 2008.

[61] 'Physics engine'; http://en.wikipedia.org/wiki/Physics_engine; September 2008.

[62] 'Google Maps'; http://maps.google.com.au/; October 2008.

[63] 'Gear ratio'; http://en.wikipedia.org/wiki/Gear_ratio; August 2008.

[64] 'Proportional Control'; http://en.wikipedia.org/wiki/Proportional_control; October 2008.

[65] ‘PID Controller’; http://en.wikipedia.org/wiki/PID_controller; October 2008.

[66] ‘How to Write a Thesis: A Working Guide’; R Chandrasekhar; Australian Research Centre for Medical Engineering (ARCME); The University of Western Australia; Written: 24 Feb 2000; Revised: 30 April 2002.

Appendix

A.1 Running the AutoSim Simulation: Overview

This tutorial gives a brief overview of ways to get the simulator running and lists some important points to remember when working with the simulation. Its purpose is to provide an easy understanding of the system for beginners.

One option is to run the simulation by running the batch file for a particular simulation scenario. A scenario specifies the selection of a particular world, driving robot and user program.

An alternative method is to start the client and server separately. For this the following steps need to be followed.

1. Start the server.
2. If you would like to run the simulation in a particular environment, select the desired world file by going to the 'File' menu. Simulation will select the default world if this step is not followed.
3. Load the server side simulation by clicking on the 'Load' button or using the 'Load' option from the 'Simulation' menu.
4. Start the client.
5. Here again select the desired world file by going to the 'File' menu.
6. The desired user interface can be selected through the 'Select User Program' option in the 'File' menu or by pressing the 'Change' button available on the client interface. The interface also allows you to view the default user program selected.
7. The client user interface (UI) also allows you to specify the name of the robot car you desire to drive in the simulation.
8. Load the client side simulation, again by clicking on the 'Load' button or using the 'Load' option from the 'Simulation' menu.
9. Run the server and the client by clicking on the 'Run' button on their respective UI or using the 'Load' option from their 'Simulation' menus.

Some Points to Remember:

- Make sure that the same world file is selected on both the server and the client.

- When specifying the name of the desired robot on the client, check that the robot is available in the selected world by looking at the simulation tree on the servers' user interface. It shows a list of all robots available in the loaded world.

- If a desired robot is not present in the selected world, it can be added to it by including it within the <*objects*> tag of the xml file corresponding to that world. The following is the format for adding a robot:

  <Robot name="unique name for every robot">
  <Description file="path of the xml file defining the robot" />
  <UserProgram file="user program that controls the robot on the server"/>
  <Position x="-26.0" y="8.0" z="15.0" roll="0.0" pitch="0.0" yaw="0.0"/> (Initial position of the robot in the world)
  </Robot>

A.2 Adding Actuators or Sensors to a Car Model

The steps to be followed for adding actuators or sensors to a car model are outlined here. Every car model is described in an xml file that contains information about its parts and devices.

- The <*Parts*> tag contains information of the body parts of the robot and links connecting them; and the <*Devices*> tag specifies all sensors and actuators attached to the body parts of the robot. So when adding a new sensor, it must go under the <*Devices*> tag.

- The format that needs to be followed is explained using the example of a gear device:

  (Tag name for device name)
  <gearDevice name="gear">
  (Part to which the device will be attached: from the <*Parts*> tag)
  <Part name="chassis" />
  (Other tags specifying parameters relevant to the sensor/actuator.)
  <mode value="1.0" />
  <diffRatio value="4.07" />
  </gearDevice>

- The next step is including the new devices' tag in the code for parsing the robot xml. This is done by adding the corresponding function in the '*ServerRobotBuilder*' class.

A.3 Autonomous Driving

This tutorial describes the ways to run the autonomous driving routines implemented and aims to provide a starting point to those interested in taking the implementation further by adding more features or functionality.

- The user program is currently implemented for driving around the world named '*block_test_area*'. The corresponding batch files that can be used to run the scenario which includes this world and the autonomous robot are '*autoDrive_in_block_joystick.bat*' and '*autoDrive_in_block.bat*'.

- In this world, the autonomous robot is positioned in such a way that it is visible in front of the user driven robot car, when the simulation loads. Note that the camera is positioned with respect to the user driven vehicle and hence the user needs to drive behind the autonomous car to keep it in sight. However, if the user decides to drive elsewhere, the autonomous car will still keep going around the block; but now we cannot track its motion.

- This user program runs on the server and hence needs to be specified in the world file under the *<UserProgram file>* tag of the robot that will be controlled by it. Currently, the impreza car is being controlled.

- The latitude and longitude difference values used in detecting intersections at which the robot will turn are specific to the '*block_test_area*' world and will need to be changed for running in other worlds. However, the same values can be used in the *'residential_area'* world, as all dimensions of these two world maps are same, except that the *'residential_area'* world has additional details like houses, traffic lights and trees.