OBSTACLE AVOIDANCE CONTROL FOR AUTONOMOUS GROUND VEHICLE

By

M. Ahsan Nisar
(NUST201361569MPNEC45013F)

PAKISTAN NAVY ENGINEERING COLLEGE, PNS JAUHAR, KARACHI

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

August, 2016

OBSTACLE AVOIDANCE CONTROL FOR AUTONOMOUS GROUND VEHICLE

By

M. Ahsan Nisar
(NUST201361569MPNEC45013F)

REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MS

TO THE FACULTY OF POSTGRADUATE PROGRAM

PAKISTAN NAVY ENGINEERING COLLEGE, PNS JAUHAR, KARACHI

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

August, 2016

ABSTRACT

This work focuses on developing an obstacle avoidance algorithm for autonomous hybrid electric vehicles named as Obstacle Avoidance System (OAS). The algorithm is subdivided into two parts, i.e. heading angle and velocity adjustment. A pre-requisite to OAS is an obstacle detection system, which provides characteristics of obstacles. A nascent approach of trajectory comparison and estimation has been proposed. A potential point of impact is estimated from trajectory of each obstacle and requisite heading and velocity corrections are provided to the controller. The performance of different operational scenarios is evaluated by simulations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	***Notation***
CF	Cost Function
$\emptyset_{gap_c}$	Gap center angle
D	Distance between vehicles
x	Position of Autonomous car at x-axis
y	Position of Autonomous car at y-axis
x_0	Initial position of the car along x-axis
y_0	Initial position of the car along y-axis
δx	Distance between car and no go zone
$\mathrm{y}_{\mathrm{car}}$	Current position of car along y-axis
x_{car}	Current position of car along x-axis
x_{obs}	Current position of obstacle along x-axis
y_{obs}	Current position of obstacle along y-axis
θ_{req}	Required heading angle

Symbol	***Notation***
A	Constant of proportionality
x_f	Final position of car along x-axis
y_f	Final position of car along y-axis
t_i	Initial time
t_f	Final time
δt	Change in time
δy	Distance between lanes
v	Velocity
V_{car}	Velocity of the car
V_{obs}	Velocity of the obstacle
$\delta_{x_{obs}}$	Change in position of obstacle along x-axis
$\delta_{y_{obs}}$	Change in position of obstacle along y-axis
V_{req}	Required velocity
V_{drop}	Drop in velocity

Subscript	***Notation***
obs	Obstacle
fov	Field of view
req	required
r	Right side
l	Left side
x	Along x-axis or about x-axis
y	Along y-axis or about y-axis
z	Along z-axis or about z-axis
f	final
i	initial
los	Line of sight

LIST OF ACRONYMS

Abbreviation	***Acronym***
AGV	Autonomous Ground Vehicle
OAS	Obstacle Avoidance System
AHEV	Autonomous Hybrid Electric Vehicle
APF	Artificial Potential Field Method
FGM	Follow the Gap Method
VFH	Vector Field Histogram
VSBM	Vision Sensor Based Method
HNA	Hybrid Navigation Algorithm
NHNA	New Hybrid Navigation Algorithm
IPA	Impact Point Algorithm
OAS	Obstacle Avoidance System
FB	Feedback
GPS	Global Positioning Sensor

Abbreviation	***Acronym***
Cdr	Commander (Pak Navy)
HEV	Hybrid Electric Vehicle
Dr.	Doctor (PHD)
INS	Inertial Navigation System
NUST	National University of Science & Technology
PNEC	Pakistan Navy Engineering College
RADAR	RAdio Detection And Ranging
LIDAR	LIght Detection And Ranging
SONAR	SOund Navigation And Ranging

ACKNOWLEDGEMENTS

All praise and thanks to the Almighty ALLAH (SWT) whom the help and guidance is sought for sustenance, who gave me the courage and knowledge to undertake this intriguing as well as interesting project.

I am greatly indebted to my thesis supervisor, Dr. Cdr Attaullah, for his immense guidance, suggestions and wise reproach throughout the course of the thesis. It was his appreciation which enhances my efficiency in completing the project and made me able to design the Obstacle Avoidance System for driverless ground vehicle.

Last but not least, I shall always remain highly obliged to my parents for giving me immense strength not only to set high goals but also to achieve them. Without their support, I would never have gotten so far.

DEDICATION

To Humanity.

CHAPTER 1 : INTRODUCTION

Introduction

The advancement of electronic era has made human life easier as compared to last century. Humans are inventing systems/machineries, which are capable of performing human tasks to ease human life, replace labour, do recursive tasks and work in life threatening environment. Along with these inventions, humans started taking advantage of electronic inventions more efficiently to save their time, do parallel tasking and speed up their work. These days world is using autonomous systems for travelling, research, exploration, military purpose, household use, industrial use, offices use and many other uses for better efficiency.

Since auto pilot systems have been serving humans to travel in air and sea for decades and during these years auto pilot systems have become the basic need of aviation and marine industries. Similarly, autonomous systems to drive vehicle on the road would be the need of coming decades to serve humanity. Keeping this need in mind many research organizations and research institutes are working on it and this is the basic motivation to work on this topic for a thesis.

A driverless car is a robotic vehicle that is designed to reach to destinations without a human driver. Autonomous vehicle must be able to navigate without human intervention to a predetermined path.
This work is a part out of three for development of complete model and subsequent controller for fully autonomous hybrid electric vehicle that drives it in the safest way possible, avoids the obstacles intelligently and take the decision instantly.

Obstacle Avoidance System

Like any other vehicle, Autonomous vehicle also works in a systematic way. It needs a navigation system for path information and the path trajectory system for lane keeping. Then comes the obstacle detection system. While following the path obstacle detection system detects obstacle and send the information to an obstacle avoidance system which generates the required heading and velocity and drive control act upon the decision and the car runs smoothly and reach to the destination without collision.

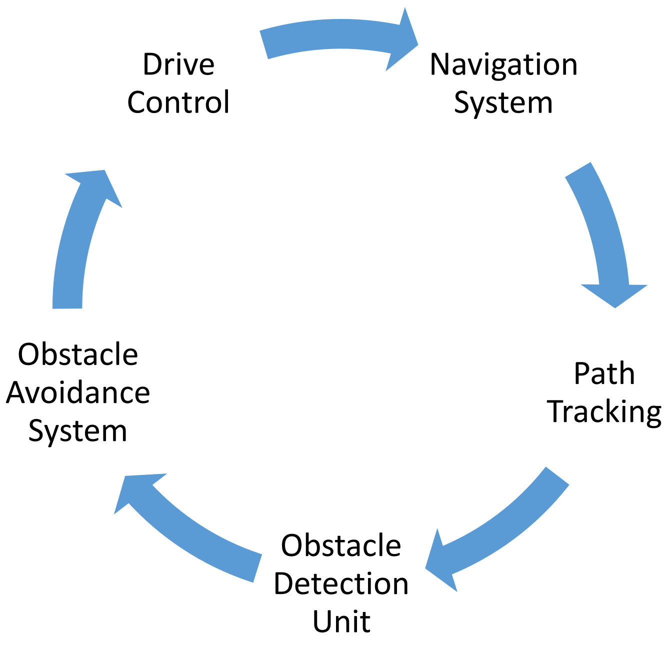


Figure 1 Major Units of Autonomous Car

Obstacle Detection Unit

Obstacle detection is the key capability for an autonomous vehicle systems, without the obstacle detection unit, the concept of autonomous vehicle has no meaning. For the obstacle detection and then avoidance, it is very much necessary to categorize the peculiar threat and deals them accordingly.

An autonomous Vehicle should be able to interact with the environment. For this purpose many sensors are being used, some majors are mentioned below:

I. RADAR

The radars are installed on front and back side of the car and on the sides of the bumper for the adaptive cruise control. It senses the object from approximately 300 meters and accidents prevention system trigger when the output is true.

II. LIDAR

Lidar is abbreviated as Light Detection and Ranging. It acts as an eye of autonomous car and generates an accurate map of the car’s surroundings, but it is not ideal for monitoring the speed of other cars in real time [1]. It consists of an emitter, Mirror and Receiver. The Emitter sends out the laser beam which bounces off the mirror that is rotated along with the cylindrical housing at 10rpm. The bounced beam returns to mirror and then received by the receiver.

III. CAMERA UNIT

The camera unit is mounted on an autonomous car as per the field of view of the camera. Cameras which are used by the google cars, have 50^0 field of view and mounted on the exterior of the car in pairs with small separation between them [1]. The cars knows the exact location of the obstacle as long as it is detected by many cameras. The camera unit is particularly used for the cautionary signs, pedestrian, animals and other moving objects.

IV. ULTRASONIC SENSORS

These are short range sensors. First time in history of autonomous cars, the sensors are used by Tesla autonomous cars. They are used for auto parking systems and as a backup for RADAR system. They detect everything small child, running dog etc. [2]

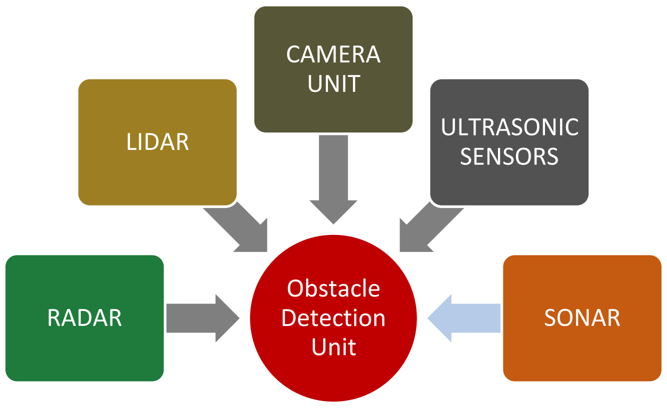


Figure 2 Sensors for Obstacle Detection

Challenges

The development of self-driving cars has been an area of active research for many years. Recently, an abundance of interest has been engendered by Google's self-driving cars, which have logged many thousands of miles in numerous scenarios – city driving as well as highway driving, to name a few. While there have been a couple of mischances, the reason for these mishaps have been the people driving different vehicles, which have slammed into the self-driving auto. A self-driving car by itself has never been involved in any mishap initiated by it. While the technology behind self-driving cars appears to be incredible; it is yet to confront numerous genuine difficulties. One of the difficulties is that of driving where legitimate roads don't exist. Another challenge lies with the execution of these vehicles in amazing climate conditions like overwhelming precipitation and/or substantial snow.

Given that the technologies being used will mature over time, it is only a matter of time, where the performance of self-driving cars will surpass the expectations of any vehicle in the city. Even with the maturity of the technology behind self-driving cars, we believe that the wider acceptance of such vehicles needs to address three important issues, namely sensing technology interference, accident liability responsibility and crowd navigation.

Problem Statement

To build an Obstacle Avoidance System for two wheel equivalent model of Autonomous Ground Vehicle by presenting a reference signal for speed and heading of vehicle in order to avoid stationary and moving obstacles on a highway, assuming that obstacle is already detected by obstacle detection unit.

Advantages

Significant benefits of AGVs in general are:

(a) Accident avoidance
(b) Fewer road traffic collision
(c) Better roadway capacity by reduced stream of traffic crowding.
(d) Assistance of automobile occupants from driving and navigation tasks.
(e) Human error reduction.
(f) Better mobility.
(g) Reduction in requirement of traffic police

Major drawbacks / disadvantages

Along with lots of benefits there are potential disadvantages of AHEVs enlisted as:

(h) High cost of vehicles due to sensitive and state of the art hardware.
(i) Weather condition may effect performance.
(j) Increased number of accidents till maturity of system and technology.

(k) Raise of unemployment due to self-driving systems (less need of drivers) as we shift from cars and heavy traffic to self-driving tractors and machinery.

Report Layout

Chapter 1 gives a brief introduction on autonomous cars. The major phases of building autonomous vehicle have been discussed. Sensors of obstacle detection unit has been briefed and obstacle avoidance system is discussed. And at the last some advantages and challenges have been enlisted.

Chapter 2 covers the literature review of obstacle avoidance system with respect to the technology and work done in local as well as global market. It briefly describes the major algorithms used by the researchers so far.

Chapter 3 describes the major contribution of obstacle avoidance system. This chapter includes the designing IMPACT POINT ALGORITHM, its mathematical derivation and modelling. It also includes modelling of some major cases of obstacle avoidance system. In last graphical user interface of the algorithm has discussed.

Chapter 4 is the brief chapter of simulations and results of derived cases of the algorithm. Result has been analysed and compared with already existed algorithms used by different major companies working on autonomous systems.

Chapter 5 concludes the research work.

Chapter 6 discusses the future work in order to make the algorithm more robust of the most complex environment.

CHAPTER 2 : LITERATURE REVIEW

Introduction

After Obstacle recognition the principle undertaking for the driverless vehicle is to maintain a strategic distance from that obstacle. There are various Algorithms through which an obstacle can be maintained a strategic distance from. These algorithms were at first intended for versatile robots yet they are executed on AGVs also.

Diverse configuration algorithms have been introduced so far to talk about the possibility of obstruction evasion, which incorporate the Vector Field Histogram Technique, Follow the Gap Method, Vision Sensor based Method, Hybrid Navigation Algorithm with Roaming Trials and New Hybrid Navigation Algorithm. [3], [4]

Algorithms

Follow the Gap Method

Follow the gap method dodges obstacles by finding the gap between them. It calculates the gap angle. It has a threshold gap, the smallest gap between obstacles from which vehicle can move. If the measured gap is greater than the threshold gap then vehicle will follow calculated gap angle. Obstacle avoiding using “FGM” is done in three main steps. [5]

STEP-1: Calculate the gap array and finding the maximum gap

In step 1, when vehicle face obstacles it calculates the distance of obstacle from vehicle and stores these distance in distance array. After finding the distances of all obstacles, gap array is generated, which includes the gap between obstacles. Gap array is being traversed to find a maximum gap between obstacles. If more than one Maximum gap exists with the same value, then first gap will be selects as a maximum gap. The pink lines are indicating the non-holonomic constraints of vehicle whereas doted green lines are the field of view of vehicle. d_{nhol_l} and d_{nhol_r} are the distances of obstacles from left and right non-holonomic constraints lines and d_{fov_l} and d_{fov_r} are the distances of obstacles from left and right field of view lines respectively. $\emptyset_{nhol\_l}$ and $\emptyset_{nhol\_r}$ are the angles of left and right non-holonomic constraint lines and $\emptyset_{nfov\_l}$

and $\emptyset_{nfov\_r}$ are the angles of left and right field of view lines of cars. The distance with less value is stored and avoided first. [3]

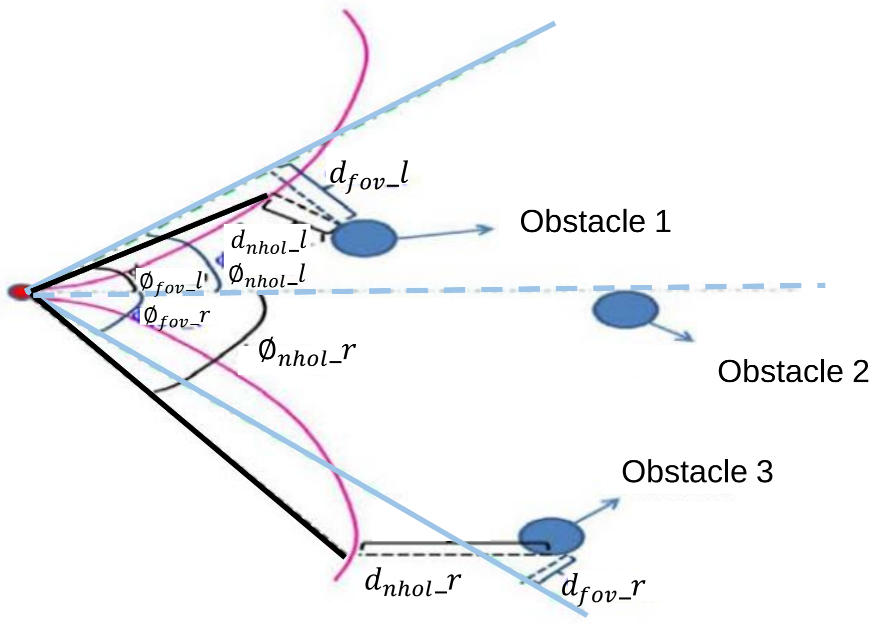


Figure 3 Representation of gap border parameter [3]

STEP-2: Calculation of gap center angle

The second step of FGM is to calculate center angle of the maximum gap, which ensures the safe trajectory from the center of obstacles. This is an angle of vector having tail at vehicle's current position and head on the center point of maximum gap. Gap center angle can be calculated by using Apollonius theorem and law of cosine. Final equation of gap center angle is; [4]

$$\emptyset_{gap_c} = \arccos(\frac{d1 + d2\cos(\emptyset_1 + \emptyset_2)}{\sqrt{d_1^2 + d_2^2 + 2d_1 d_2 \cos(\emptyset_1 + \emptyset_2)}} - \emptyset_1$$

Where d_1 and d_2 are the distances of obstacle 1 and 2 from the car respectively. $\emptyset_1$ and $\emptyset_2$ are angles of obstacle 1 and 2 respectively, $\emptyset_{gap_c}$ is the final calculated gap center angle.

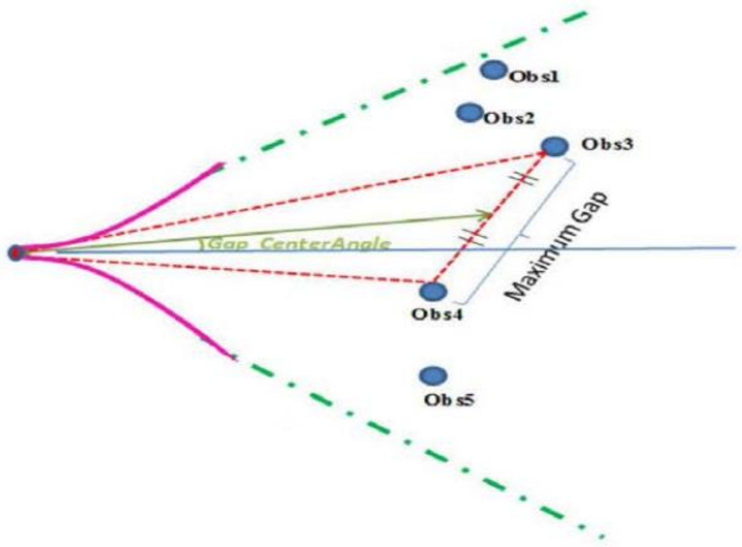


Figure 4 Representation of gap center angle [3]

STEP-3: Calculation of final heading angle

The last stage of “FGM” is to calculate the final heading angle. This can be achieved by combining the gap center angle with the goal angle. The combining structure is distance of obstacles and weight dependent, i.e. the obstacle nearer to vehicle has more weight. In case, when obstacle is at very short distance to car then vehicle must move to gap angle rather than goal angle. It is due to fact that obstacle avoidance is the main task of path planning. [4] [3]

$$\emptyset_{final} = \frac{\frac{\alpha}{d_{min}}\emptyset_{gap_c} + \beta\emptyset_{goal}}{\frac{\alpha}{d_{min}} + \beta}$$

Where $d_{min} = \min_{i=1:n}(d_n)$, $\emptyset_{gap_c}$ and $\emptyset_{goal}$ are calculated gap and goal angle, α and β are weight coefficient of gap and goal angle respectively. [3]

New Hybrid Navigation Algorithm

New hybrid navigation algorithm based on two layers, deliberative layer and reactive layer. Both layers are independent to each other. Deliberative layer planned a reference path on the basis of stored prior information. Reactive layer independently steers vehicle on the path planned by the deliberative layer. [6]

Hybrid algorithm required prior information of environment, which is stored in the form of binary grid map. In map, states of every grid are either free of occupied that depends on obstacles around i.e. free for no obstacle and occupied for obstacle. Unknown information is also taken as a free. In deliberative layer, a reference path is generated. Reference path is temporary and not necessary to follow through out motion, it can be changed by the reactive layer.

Reactive layer takes reference path from deliberative layer and controls the motion of robot. It also receives the precepts of sensors and take decision to avoid an obstacle if found. For the purpose of obstacle avoidance, this layer uses D-H bug algorithm (Distance Histogram bug). This is a version of bug-2 algorithm, which allows vehicle to rotate freely at angle less than 90° to avoid an obstacle. If the rotation of 90° or greater is required to avoid an obstacle; it acts as bug-2 algorithm and starts moving to destination when path is clear from obstacles.

Reactive layer can change the path on the basis of current percept. Sensors provide current precepts to reactive layer as well as it updates the prior knowledge. In case on conflict between layers, the result of reactive layer is taken into an account. It is due to the present and updated nature of the results of reactive layer and hence incomplete knowledge of deliberative layer may contain errors. [6]

Hybrid Navigation Algorithm with Roaming Trials (HNA)

The Hybrid Navigation algorithm with roaming trails is related to new NHNA. The main difference is that it used APF instead of D-H BUG in reactive layer. NHNA has not described any limit for car to deviate from reference path but HNA used the concept of roaming trails for the same purpose.

Comparison of Commercial Vehicle

Google Self Driving car collects data from different sensors mounted on it. The software processes all of the data in real-time as well as modelling behavioural dynamics of other drivers, pedestrians, and objects around you. While some data is hard-coded into the car, such as stopping at red lights, other responses are learned based on previous driving experiences. Every mile driven on each car is logged, and this data is processed in an attempt to find solutions to every applicable situation.

The learning algorithm processes the data of not just the car you're riding in, but that of others in order to find an appropriate response to each possible problem. [1]

Tesla's forward looking camera, created by Mobileye, is mounted on the front of the rear-view mirror and features advanced software that allows it to not only measure distance, but also read signs, and detect pedestrians and avoids it. [2]

Toyota's Pre-Collision System packs in a similar combination of radar and cameras to help its latest cars avoid accidents at all costs, with the technology splitting potential accidents into four distinct steps: The detection of a vehicle ahead, the possibility of a collision, the high possibility of a collision and collision is unavoidable. During these steps the car will offer an alarm and visual warning, followed by braking assistance and finally fully automatic braking. Toyota car only avoids the obstacle by apply brakes.

Volvo's avoidance system offers audible and haptic warnings, only activating auto braking when it has to, and then full braking power a second before impact. Even if that's not quite enough to stop you from crashing altogether, it's likely to reduce your speed by a considerable amount, potentially proving the difference between a life-threatening crash and a much smaller collision.

Mercedes-Benz has also joined in the action, introducing Collision Prevention Assist, which uses radar to continuously monitor the distance to the vehicle in front. Driver will get several seconds to respond to a warning light in the instrument cluster if you get that little bit too close to the vehicle in front, with an intermittent tone sounding if the distance decreases quickly. If braking by the driver isn't enough to slow the car sufficiently, Collision Prevention Assist will also apply the brakes.

BMW as one of the leading car companies when it comes to technology, it's no surprise to see BMW has been working on its own collision-avoidance technology, in this case dubbed Active Assist, and future BMWs are set to be a whole lot smarter than the average car.

Active Assist uses a huge range of laser scanners, cameras and even a few ultrasound sensors that will make your journey safer by detecting obstacles and other vehicles in the immediate surroundings. Four laser scanners can measure precise distances to other objects, along with measuring speeds and surroundings, and BMW explains that the car can then form a picture of which areas are accessible and free from obstacles.

Most common machine learning algorithms that are being used in autonomous vehicles are based on OBJECT TRACKING. These algorithms are aimed at improving the accuracy of pinpointing and distinguishing between objects. A core problem, for example, is profiling of an object i.e. whether is it another vehicle, a pedestrian, a bicycle, an animal? The solution is a sophisticated machine learning or pattern recognition algorithm that is fed with many images containing objects. [7]

A vehicle’s internal map includes the current and predicted location of all static (e.g. buildings, traffic lights, stop signs) and moving (e.g. other vehicles and pedestrians) obstacles in its vicinity. Obstacles are categorized depending on how well they match up with a library of pre-determined shape and motion descriptors.

CHAPTER 3 : SYSTEM MODELLING AND ALGORITHM DESIGNING

System Modelling for Obstacle Avoidance

An equivalent two-wheel model for vehicle dynamics is considered. Where theta is the car heading angle and phi is the steering angle of the vehicle. It has been given in the literature [8] that the relation of car's heading with steering is:

$$\dot{\theta} = \frac{v}{l} \tan \varphi$$

Where v is constant velocity of vehicle and l is the distance between front and rear tires.

The author does not agree with the designed model and hence a new model right from the start is developed. Consider Newton's 2nd law of motion in angular motion

$$M_z = I\,\ddot{\theta}$$

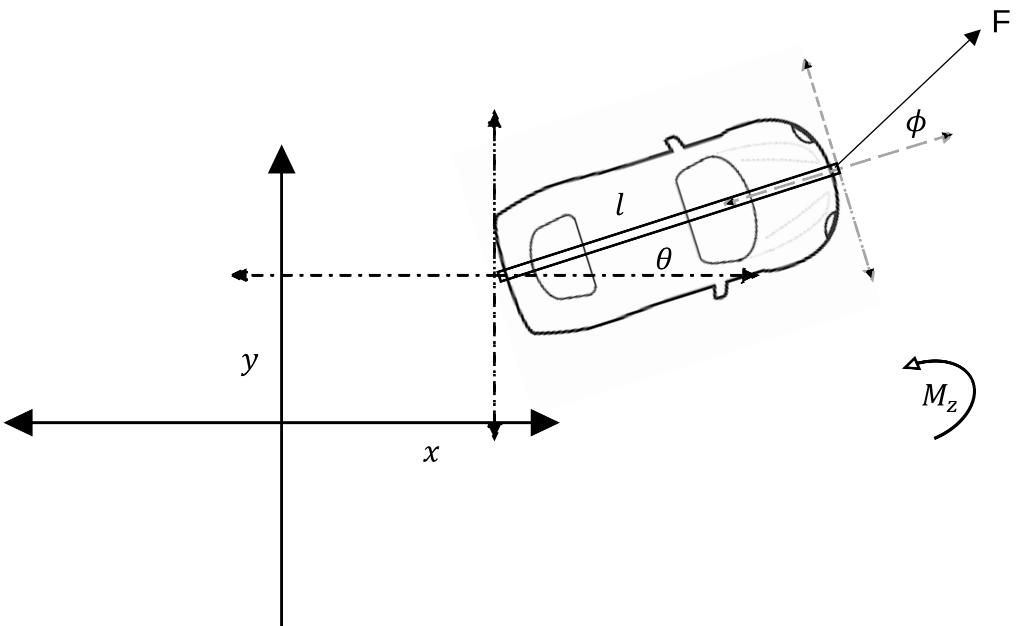


Figure 5 Vehicle Two Wheel Equivalent Model for Steering Motion

It can be acquired from the figure above that the force acting along the line of the vehicle longitudinal axis has no effect on vehicle steering motion, whereas the lateral acting force steers the vehicle. Hence:

$$M_z = l\,F\sin\phi$$

Comparing for M_z

$$I\,\ddot{\theta} = l\,F\sin\phi$$

By applying Newton's 2nd law of motion in translational form:

$$F = m\,a$$

$$F = m\frac{dv}{dt}$$

$$\boldsymbol{I\,\ddot{\theta} = l\,m\,\frac{dv}{dt}\sin\phi} \qquad \textbf{3.1}$$

$$\ddot{\theta} = \frac{l\,m}{I}\frac{dv}{dt}\sin\phi$$

We know that for a rigid body as shown in the figure above the moment of inertia is defined as:

$$I = m\,l^2$$

$$\ddot{\theta} = \frac{1}{l}\frac{dv}{dt}\sin\phi$$

Integrating both sides over time

$$\int \ddot{\theta} = \int \frac{1}{l}\,\frac{dv}{dt}\,\sin\phi$$

$$\dot{\theta} = \frac{1}{l}\int \frac{dv}{dt}\,\sin\phi$$

Simplification of the product of a function in the integral it is calculated that:

$$\int \frac{dv}{dt}\,\sin\phi = v\sin\phi$$

Therefore

$$\boxed{\dot{\theta} = \frac{v}{l}\sin\phi} \qquad 3.2$$

The equation (3.2) shows a relation of heading angle of vehicle with the steering angle and the velocity as a direct relation. Here it is to be noticed that the time-varying variables $\theta(t), \phi(t)\ and\ v(t)$ are written in simpler form as $\theta, \phi\ and\ v$ respectively. Let us consider velocity as constant to focus on steering controller:

$$v = constant$$

In order to design the equation of motion for vehicle steering in state space form, let the system states as heading and steering angles of vehicle, whereas the output of the closed loop system is desired as the heading of vehicle. And the control effort to correct heading is the steering angle velocity. The states as defined above can be written as:

$$x_1 = \theta(t)$$

$$x_2 = \phi(t)$$

Control is applied on rate of change of steering angle

$$u = \dot{\phi}(t)$$

Heading angle is considered as output of the system

$$y = \theta(t)$$

Therefore the system becomes

$$\dot{x_1} = \dot{\theta} = \frac{v}{l}\sin\phi$$

$$\dot{x_1} = \dot{\theta} = \frac{v}{l}\sin x_2$$

$$\dot{x_2} = \dot{\phi} = u$$

Which can be summarized as

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \dot{\varphi} \end{bmatrix}$$

$$\dot{x} = \begin{bmatrix} \frac{v}{l}\sin x_2 \\ u \end{bmatrix}$$

Whereas

$$y = x_1$$

Comparing with generic state space form i.e.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{t}, \boldsymbol{x}, \boldsymbol{u})$$
$$y = h(t, x, u)$$

Here controlled system is:

$$f(t,x,u) = \begin{bmatrix} f_1(t,x,u) \\ f_2(t,x,u) \end{bmatrix} = \begin{bmatrix} \frac{v}{l}\sin x_2 \\ u \end{bmatrix}$$

$$\boldsymbol{h}(\boldsymbol{t}, \boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x_1}$$

The system analysis shows following:

(a) Open loop plant model is **time invariant**.

(b) Plant is **nonlinear**.

(c) Output of system is not directly linked with control effort.

(d) f is continuously differentiable in domain D_x and D_u (containing origin $x = 0, u = 0$

Algorithm Designing for Obstacle Avoidance

The foremost step required to accomplish the designing of Obstacle Avoidance System is to divide the system into major subsystems. Then, as a second step a general algorithm for the flow of work for each subsystem has to be developed. Figure 6 below shows main algorithm of OAS in which prediction of vehicle trajectory has been designed.

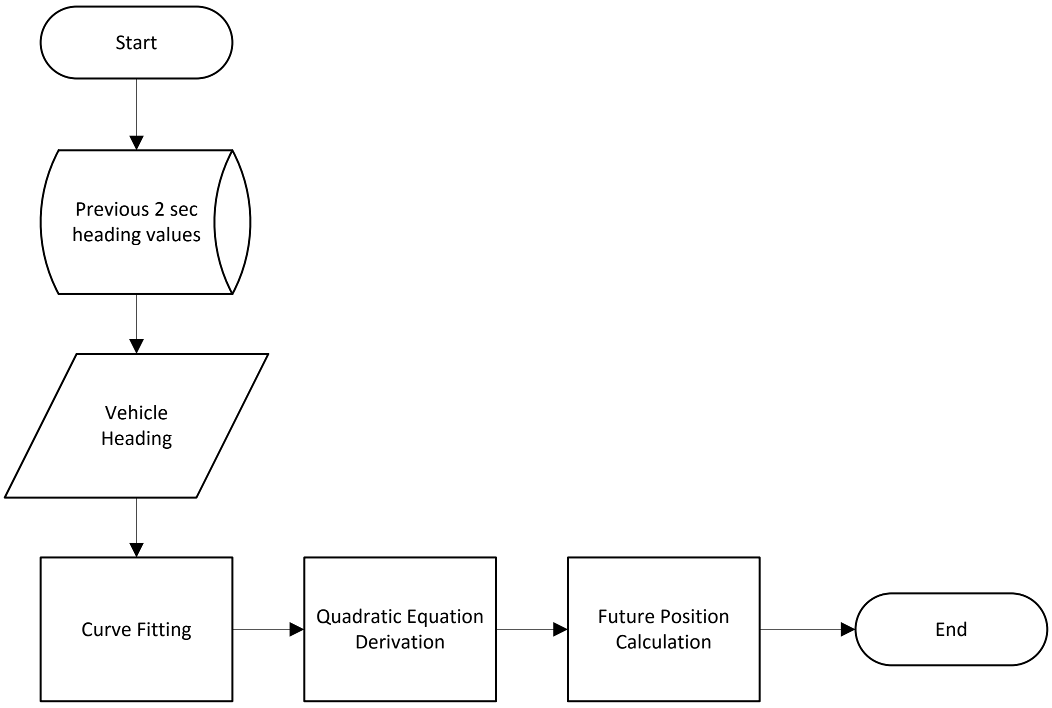

Start
Previous 2 sec heading values
Vehicle Heading
Curve Fitting
Quadratic Equation Derivation
Future Position Calculation
End

Figure 6 Flow Chart for Trajectory Development

Impact Point Algorithm

The Impact Point Algorithm has been designed to work on the basis of trajectories along with heading angle. The vehicle has been equipped with indigenously developed trajectory prediction system as shown in figure 7 where the previous tracks have been used to fit a polynomial curve and subsequently deriving the equation of the fitted curve. The trajectory equation has been used to predict the future trajectory of the vehicle. Impact point algorithms detect the moving obstacle and checks its heading and speed from the data given by different sensors. The algorithm calculates the trajectory of each vehicle and checks for point of collision. Obstacles are categorized according to their speed and their heading.

Predictive Modelling

Predictive modeling uses mathematical and computational methods to predict an event or outcome. A mathematical approach uses an equation-based model that describes the phenomenon under consideration. The model is used to forecast an outcome at some future state or time based upon changes to the model inputs. The model parameters help explain how model inputs influence the outcome. [9] [10]

The impact point algorithm predicts the point of collision based on the rules of predictive modelling. The approach predicts the future position of the obstacle by considering the equation of the obstacle trajectory. This approach helps the obstacle avoidance system in estimating the point of impact, which is avoided by the lane change and velocity management.

To illustrate the predictive modelling in a better way, below are the equations of trajectories of two different cars, based on these current equations, their future position and point of impact has been estimated and avoided by the impact point algorithm.

Equation of the vehicle’s trajectory is as follows:

$$Trajectory1 = -0.08125x^3 + 1.15x^2 - 3.475x + 6 \tag{3.3}$$

The second car's trajectory is given as under:

$$Trajectory2 = -0.7667x^3 + 12.85x^2 - 69.38x + 125 \tag{3.4}$$

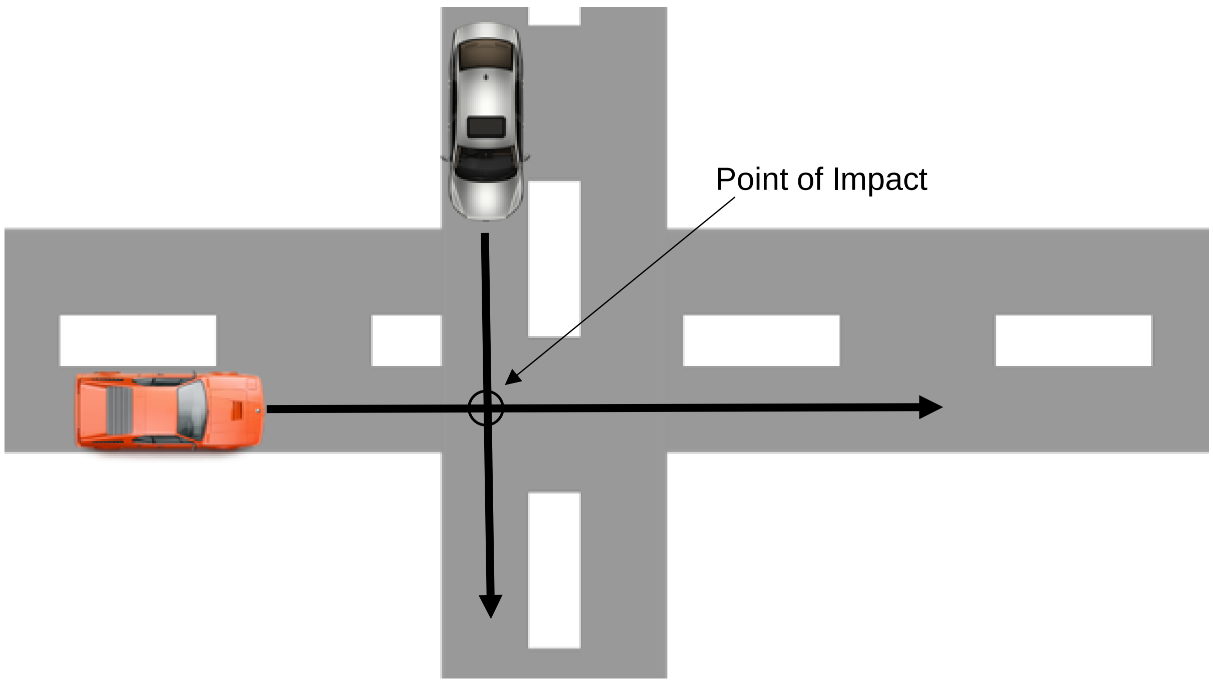


Figure 7 Estimating Point of Impact

From the given trajectories of the vehicles, the distance from the point of impact has been calculated. The distance of the trajectory1 is calculated as

$$dist1 = 2.9398m$$

The distance of the trajectory2 from the point of impact is

$$dist2 = 27.5690m$$

The time each car will take to reach at that point of impact with the velocity of 10m/s is 0.2940s for the first car, and for the second car it will take approximately 2.7569s. The amount of time is to reach at point estimated point of collision is not same, autonomous vehicle will not take the second car as a threat because it will reach to the point way before the second car reach there and cause collision.

This trajectory calculation helps the car to prioritize the obstacle and take the smart decision accordingly.

Stationary Obstacle Avoidance

To autonomously avoid the obstacle, obstacles have been categorized as stationary and moving. The algorithm developed is displayed as in below given flow chart, where sensors give the obstacle profile and once it is detected as threat impact point algorithm calculates the heading angle and avoid the obstacle.

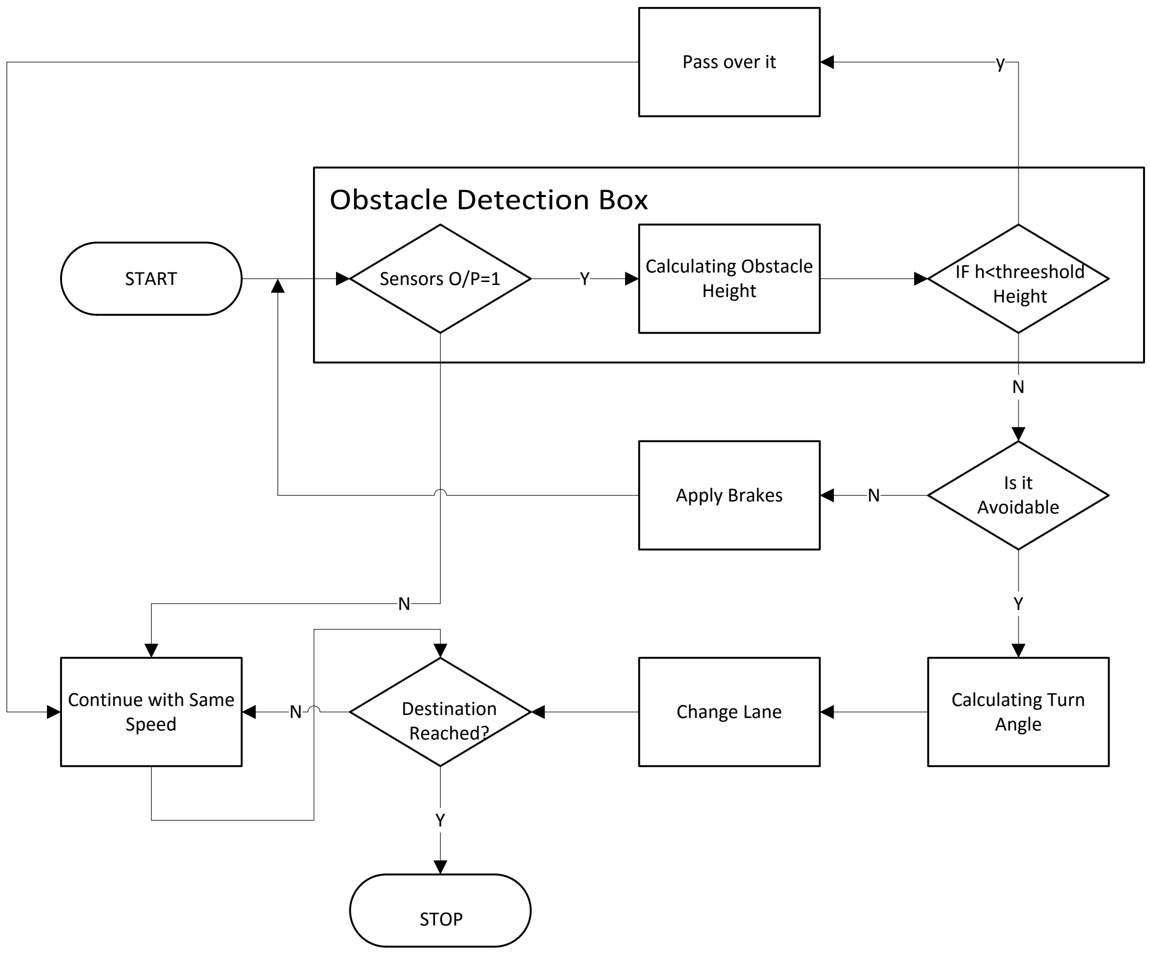


Figure 8 Flow Chart of Stationary Obstacle Avoidance

Derivation of Stationary Obstacle Avoidance w.r.t Impact Point Algorithm

In this approach, the distance between car and obstacle is measured in time not in meters. And calculating the distance from the obstacle before which car should be in another lane. Assuming the avoided distance between car and obstacle is 2 seconds, which is the international rule, to keep car apart from each other. So, in this thesis distance of 2sec is the no go zone for the car.

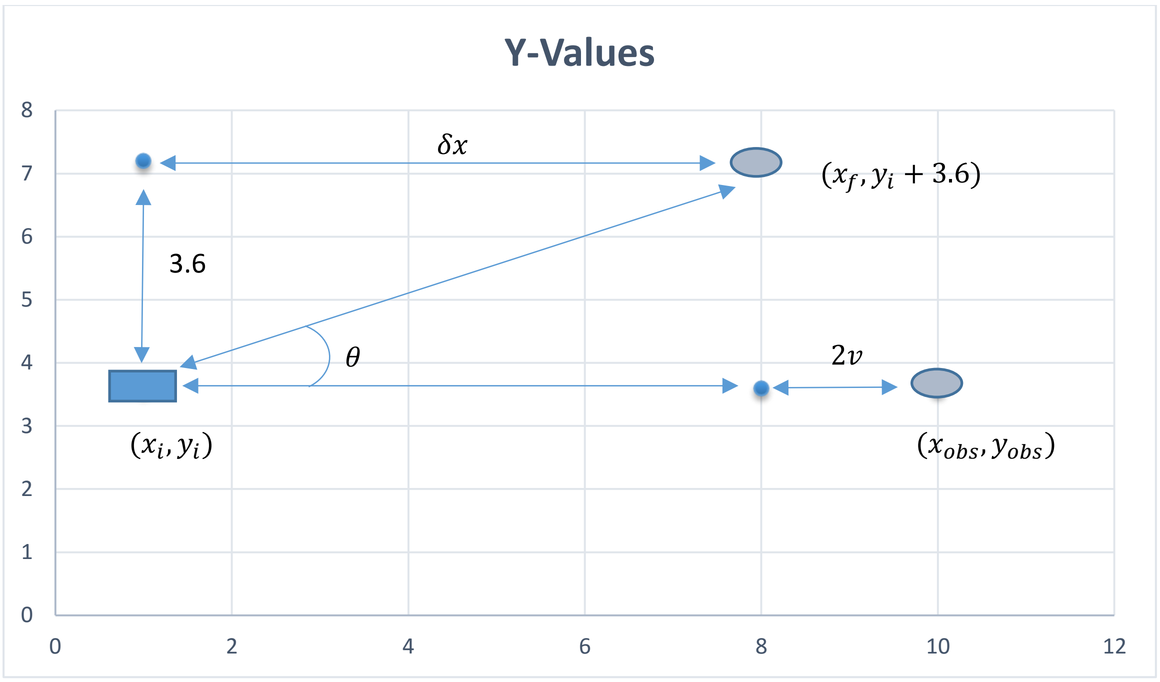


Figure 9 Stationary Obstacle Avoidance

Time to avoid the obstacle is;

$$\delta t = 2sec$$

From the distance formula;

$$\delta x = v.\delta t$$

$$\delta x = x_f - x_i$$

Assume the velocity of the car is v and the rectangular components of the velocity of the car are:

$$v_x = vcos(\theta)$$

$$v_y = vsin(\theta)$$

So the position of the car be find from above equations.

$$x = \int_{t_i}^{t_f} v_x\,.\,dt$$

$$x = \int_{t_i}^{t_f} vcos(\theta).dt$$

Similarly $y - axis$ value of the car is;

$$y = \int_{t_i}^{t_f} vsin(\theta).dt$$

Finding the final location of the car at which vehicle should be reached before the distance of $2v$, so the new location of the car is (x_f, y_f)

$$x_f = x_i + vcos(\theta).\left(t_f - t_i\right)$$

And y_f will become;

$$y_f = x_i + vsin(\theta).(t_f - t_i)$$

Where

$$\delta t = t_f - t_i$$

$$\delta x = x_f - x_i$$

By putting the values in the above equation we get the new values of δx and δy which are:

$$\delta x = vcos(\theta).\delta t$$

$$\delta y = vsin(\theta).\delta t$$

By squaring and adding above given equations;

$$\delta x^2 + \delta y^2 = (vcos(\theta).\delta t)^2 + (vsin(\theta).\delta t)^2$$

$$\delta x^2 + \delta y^2 = (v.\delta t)^2$$

And we can get the δt from above mentioned equation

$$\delta t = \frac{\sqrt{(\delta x^2 + \delta y^2}}{v}$$

By putting the value of δt in equation of δx

$$\delta x = vcos(\theta).\frac{\sqrt{(\delta x^2 + \delta y^2}}{v}$$

$$\theta = \cos^{-1}(\frac{\delta x}{\sqrt{\delta x^2 + \delta y^2}})$$

δx should be selected as per car's speed. From the very first argument we decided to give the gap between the car and obstacle in time not in meters. So, if we want a system to avoid the obstacle by 2 seconds, then the distance with respect to obstacle is;

$$distance\ w.r.t\ obstacle = 2v$$

$$distance\ of\ obstacle = x_{obs}$$

So, we have

$$x_f = x_{obs} - 2v$$

$$\delta x = x_{obs} - 2v - x_i$$

Now for the lane change, one has to keep in mind the distances between the lanes, in this system lane width of 3.6 meters has been assumed. So,

$$\delta y = L = 3.6$$

By substituting all the above values in the required heading equation, the new required heading will be;

$$\boxed{\boldsymbol{\theta_{req} = \cos^{-1} \frac{x_{obs} - 2}{\sqrt{(x_{obs} - 2v - x_i)^2 + \delta y^2}}}}$$

As the equation describes, θ_{req} depends on the position of the obstacle and car and the velocity of the car and also the distance between the lanes.

Moving Obstacle Avoidance

The idea of autonomous car came into existence to minimize the traffic accidents and the number of deaths caused by those accidents. In complex and a real environment, there are many obstacles which can cause problem for driverless cars. The most major obstacle is moving obstacle. Below is the simplified flow chart for an autonomous vehicle to avoid the moving obstacle.

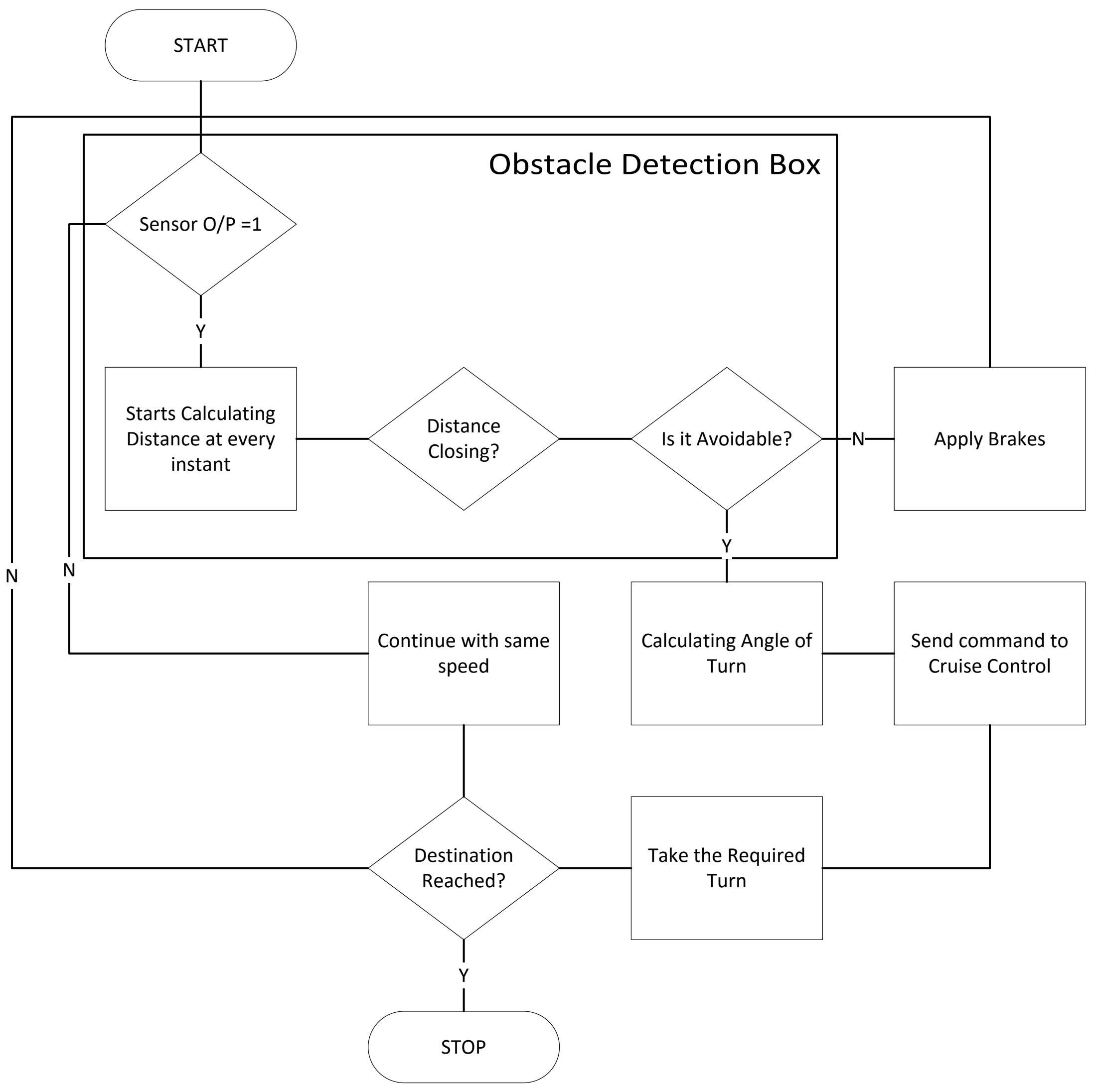


Figure 10 Flow Chart of Moving Obstacle Avoidance

Derivation of Moving Obstacle Avoidance w.r.t Impact Point Algorithm

As explained earlier, the impact point algorithm works on the trajectories. It predicts the future trajectory based on the current heading and speed and calculates the point of impact and avoid it. To depict the better picture, the flow chart is given below.

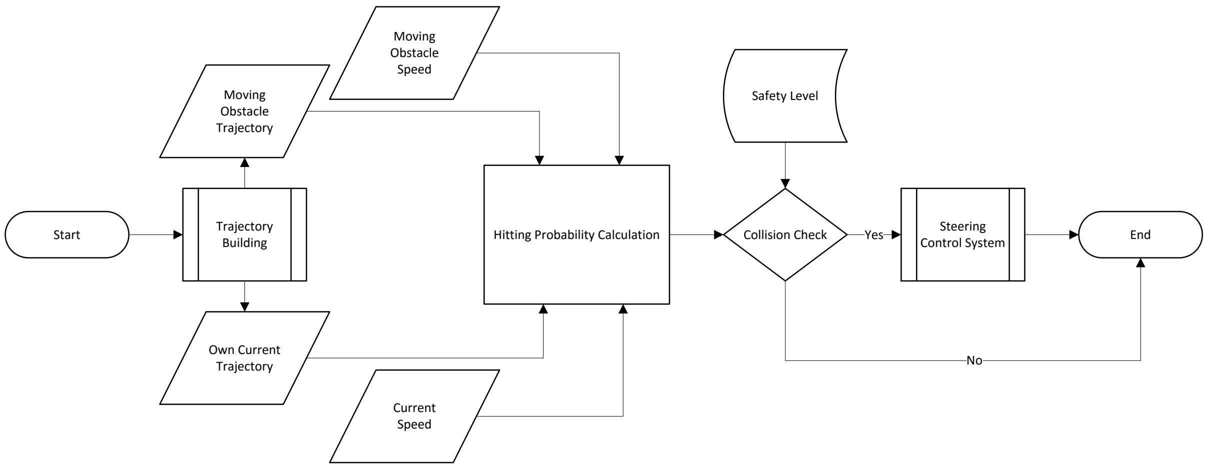


Figure 11 Flow Chart of Moving Obstacle Avoidance w.r.t IPA

θ_{req} for moving obstacle is the function of the following parameters.

$$\theta_{req} = f(x_{car}, y_{car}, V_{car}, x_{obs}, y_{obs}, v_{ob}, \delta_{xobs}, \delta_{yobs})$$

There are two approaches to avoid the moving obstacle either to increase the heading angle according to the obstacle distance and its speed or decrease the vehicle velocity depending on the position of the obstacle with respect to the car. V_{req} and θ_{req} has been derived for the moving obstacle and results are attached.

For θ_{req}, a scenario has been formulated for the two cars with different speeds and different heading angles. But the future prediction of their trajectories shows the point of collision, which should be avoided. The subject vehicle is approaching to the obstacle with greater velocity and needs a θ_{req} according to its speed to move to the next lane. The graphs of the vehicle's trajectories are given below. The prediction of the trajectories shows that there would be collision after 4 seconds.

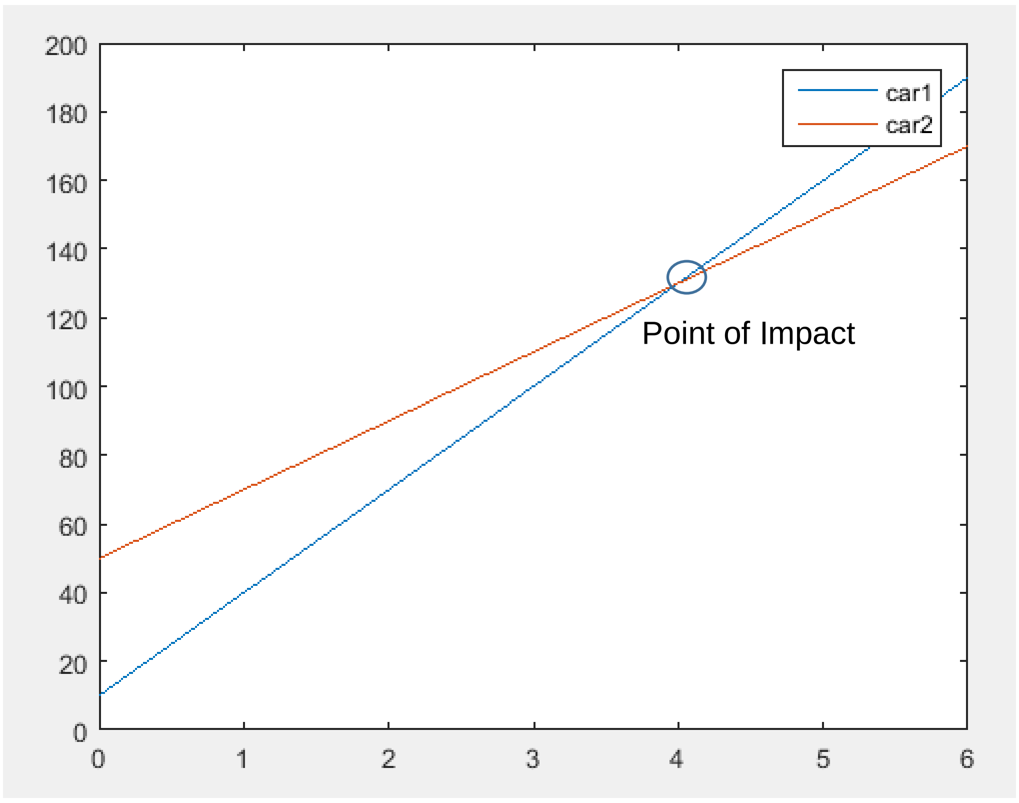


Figure 12 Trajectories of Two Vehicle with Estimated Point of Collision

Below is the equation of the above given trajectories of the two cars moving with different speed and the different heading angle is derived.

$$y = mx + b$$

$$130 = 30t + 10$$

Now calculating the time to reach at position 130, which is the point of impact for two cars.

$$t = 4sec$$

Equation of the obstacle trajectory is given below.

$$130 = 20t + 50$$

Obstacle is reaching at point 130 in 4seconds. Which is the same amount of time and thus 130 is the point of impact. Which needs to be avoided at any cost.

$$t = 4sec$$

Equating the time of impact for obstacle and vehicle will give the formulated time to reach at the impact point.

$$t = t$$

$$30t + 10 = 20t + 50$$

Replacing the numerical values with the variable.

$$v_{car} + x_i = v_{ob}t + x_{ob}$$

$$(v_{car} - v_{ob})t = x_{ob} - x_i$$

Time of impact of the moving obstacle and the subject vehicle can be calculated with the following formula.

$$t = \left[\frac{x_{ob} - x_i}{v_{car} - v_{ob}}\right]$$

We need time to avoid the obstacle. Here international rule of the traffic has been applied. Which is 2seconds distance from the front vehicle. So, time to avoid the obstacle is;

$$t = \left[\frac{x_{ob} - x_i}{v_{car} - v_{ob}}\right] - 2$$

Now, we need to find δx, which is the distance of the vehicle from the no go zone.

i.e. $\delta x = v * t$ and here t is time required to avoid the obstacle.

$$\delta x = v_{car} * \{\left[\frac{x_{ob} - x_i}{v_{car} - v_{ob}}\right] - 2\}$$

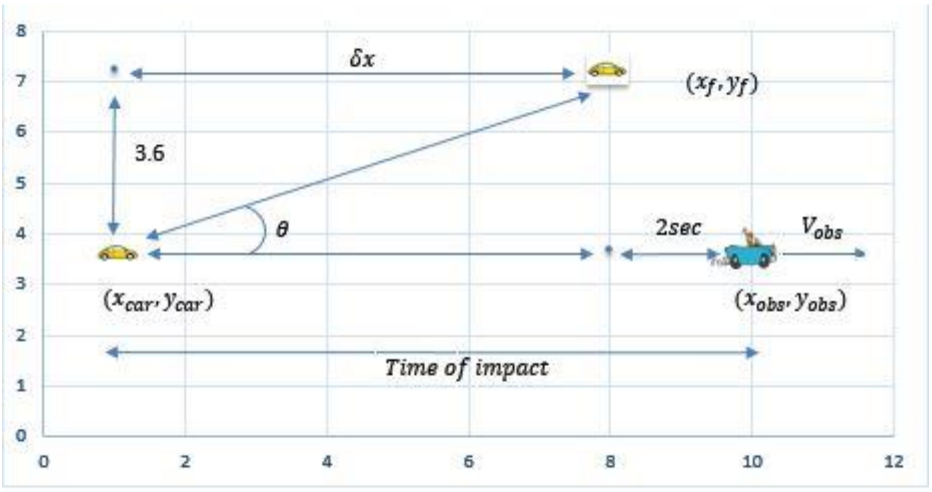


Figure 13 Moving Obstacle Avoidance by Changing Lane

Now the car will change the lane, while considering the above mentioned parameters. Lane distance is the $y - axis$ value and the δx is the $x - axis$ value of the map. So,

$$\boxed{\boldsymbol{\theta_{req} = \tan^{-1}(\frac{L}{\delta x})}}$$

Where $L = distance\ of\ lane$

Now calculating the v_{req} for the moving obstacle and the value of θ_{req}

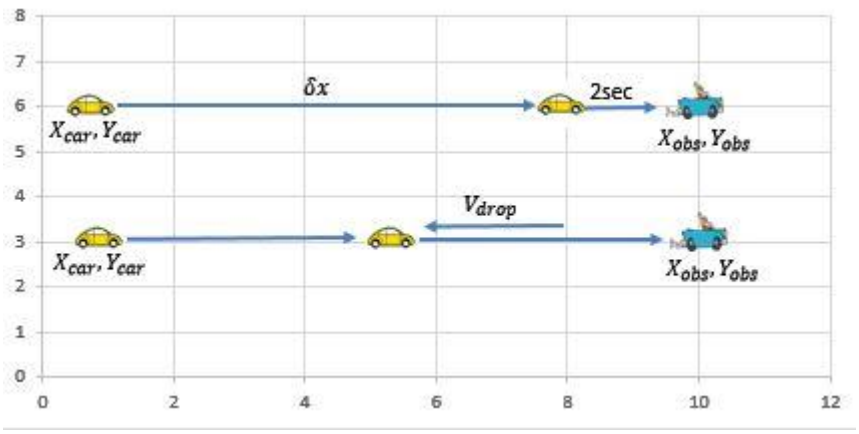


Figure 14 Moving Obstacle Avoidance by Decreasing the Velocity

To decrease the speed at $\delta x = 1m$ we need to find V_{drop}

$$\delta x = \frac{(V_{car} - V_{drop})(x_{ob} - x_i)}{V_{car} - V_{drop} - V_{ob}} - 2(V_{car} - V_{drop})$$

So we get the V_{drop}:

$$\boxed{\boldsymbol{V_{drop} = \psi + \sqrt{\frac{\alpha + \beta + \gamma}{4}}}}$$

Where;

$\psi = \delta x + 4V_{car} - 2V_{ob} + x_i - x_{ob}$

$\alpha = \delta x.(\delta x + 4V_{0b} + 2x_i - x_{ob})$

$\beta = 4V_{ob}..(V_{ob} - x_i + x_{ob})$

$\gamma = x_i(x_i - 2) + x_{ob}(x_{ob} - 1)$

V_{drop} is the amount of velocity, which needs to be decreased when the obstacle is getting closer and closer. It is the function of the velocity of the subject vehicle and the moving obstacle and their positions.

Pseudocode of Impact Point Algorithm

Initialization:

```
1. Initialize obstacle_detected to zero.
2. Input obstacle_detected from pin x.        /*reading sensor data and putting the
                                              data acquired into a variable named
                                              obstacle_detected. Pin X is arbitrary
                                              pin to be appointed as the sensor data
                                              pin (or obstacle detection circuitry
                                              output). Obstacle profile will consist of
```

```
    Obstacle Position in xy plane, Obstacle Velocity (Vx, Vy), Obstacle Heading angle $\theta$*/
3.  Set array obstacle[i] as obstacle_detected    /*Obstacle profile building in an array with i as the index of array e.g. obstacle [1] is the structure containing position, velocity and type of 1st obstacle and so on.*/
```

Obstacle Detection:

```
4.  Repeat step 2 and 3 for N times    /*minimum array size required to build the obstacle profile is N, N is the number of times the sensors give output to obstacle avoidance system to build obstacle profile */
5.  N>10    the higher limit of N depends on the completion of obstacle profile, it will stop repeating step 2 once the obstacle will not be considered as a potential threat.
6.  Build obstacle trajectory    /*by use of curve fitting tool and preparation of equation of curve for the detected obstacle. */
7.  Input vehicle[i]=accelerometer    /*Get vehicle position from accelerometer*/
8.  Build vehicle trajectory    /*by use of curve fitting tool and preparation of equation of curve for own vehicle. */
```

Impact Point Estimation

```
9.  Calculate obs_y = m1x1+c1    /*from equation in step 6. i.e. obs_y = slope_obs_traj * obs_x + c1*/
10. Calculate veh_y = m2x2+c2    /*from equation in step 8 i.e. veh_y = slope_veh_traj * veh_x + c2*/
11. POI_x = (c2-c1) / (m1-m2)    /*by equating obs_y and veh_y to find the x coordinate of point of impact*/
12. POI_y = m1*POI_x + c1    /*calculating y coordinate of point of impact by putting values in step 9*/
13. POA = (POI_x - 2*v, POI_y)    /*Point of avoidance is 2 sec before the point of impact. Here 2*v is the distance that vehicle covers in 2 sec. whereas there is no effect on y coordinate of POA*/
```

Case 1: Stationary Obstacle

```
14. Delta_x = POA – veh_x    /* Distance with the POA is an input to heading angle calculation formula*/
```

```
15. Delta_y = L                           /* L is lane to lane distance, for
                                          obstacle avoidance by lane changing.
                                          */
16. If (obstacle speed==0)                /* Stationary Obstacle Case*/
    a. Calculated Theta_req               /*$\theta_{req} = \cos^{-1} \frac{x_{obs}-2v-x_i}{\sqrt{(x_{obs}-2v-x_i)^2+\delta y^2}}$*/
    b. Output Theta_req                   /*Output reference signal to drive
                                          control system*/
```

Case 2: Moving Obstacle

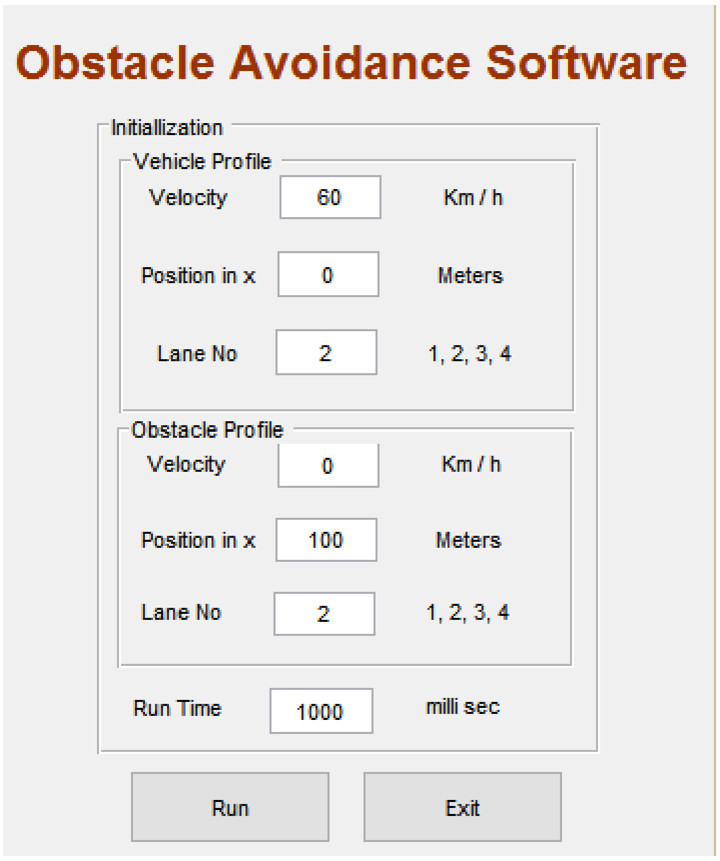
```
17. Else If (obstacle speed>0)            /*Moving Obstacle case*/
18. Build obstacle trajectory             /*by use of curve fitting tool and
                                          preparation of equation (2nd degree
                                          polynomial) of curve for the detected
                                          obstacle. The equation is used to
                                          calculate the future trajectory of
                                          obstacle, the equation is parabolic but
                                          at every instant it gives tangential
                                          heading of obstacle*/
    a. Calculated Theta_req               /*$\theta_{req} = tan^{-1}\left(\frac{\delta y}{\delta x}\right)$
                                          Where $\delta x = v_{car} * \{\left[\frac{x_{ob}-x_i}{v_{car}-v_{ob}}\right] - 2\}$
                                          and $\delta y = width\ of\ lane$        */
    b. Output Theta_req                   /*Output reference signal to drive
                                          control system*/
19. Calculate Velocity_drop               /*$V_{drop} = \psi + \sqrt{\frac{\alpha+\beta+\gamma}{4}}$;   where
                                          $\psi = \delta x + 4V_{car} - 2V_{ob} + x_i - x_{ob}$
                                          $\alpha = \delta x.(\delta x + 4V_{0b} + 2x_i - x_{ob})$
                                          $\beta = 4V_{ob}..(V_{ob} - x_i + x_{ob})$
                                          $\gamma = x_i(x_i - 2) + x_{ob}(x_{ob} - 1)$
20. Output Velocity_drop                  /*Output reference signal to drive
                                          control system*/
```

Case 3: Dead-End Scenario

```
21. While (no alternate route available)  /*i.e. all lanes occupied, traffic jam,*/
22. V_veh=V_drop                          /*V_drop is the calculated velocity
                                          depends on the obstacle profile*/
23. End
```

Impact Point Software

Matlab based graphical user interface has been developed for the clear picture of the algorithm. Different scenarios have already been tested on the software and results have been analysed in chapter 4.


Obstacle Avoidance Software

Initiallization

Vehicle Profile

Velocity	60	Km / h
Position in x	0	Meters
Lane No	2	1, 2, 3, 4

Obstacle Profile

Velocity	0	Km / h
Position in x	100	Meters
Lane No	2	1, 2, 3, 4

Run Time	1000	milli sec

Figure 15 Software of Impact Point Algorithm

Software takes obstacle profile as input and based on the mathematical equations, it calculates the avoidance angle and the desired velocity and generate its graph. User can change the position velocity and lane of the subject vehicle and change the category of the obstacle from stationary to moving by introducing its velocity.

Comparison

1. The obstacle avoidance system has been presented [8] based on fuzzy logics and optimal control. Author has successfully achieved the task but the approach is not suitable for more than three obstacles and for the simulation time greater than 15 seconds. Simulations have been tested with four obstacles to check the robustness of the system but system got crashed. Moreover, the system is useless for the unavailability of the lane.

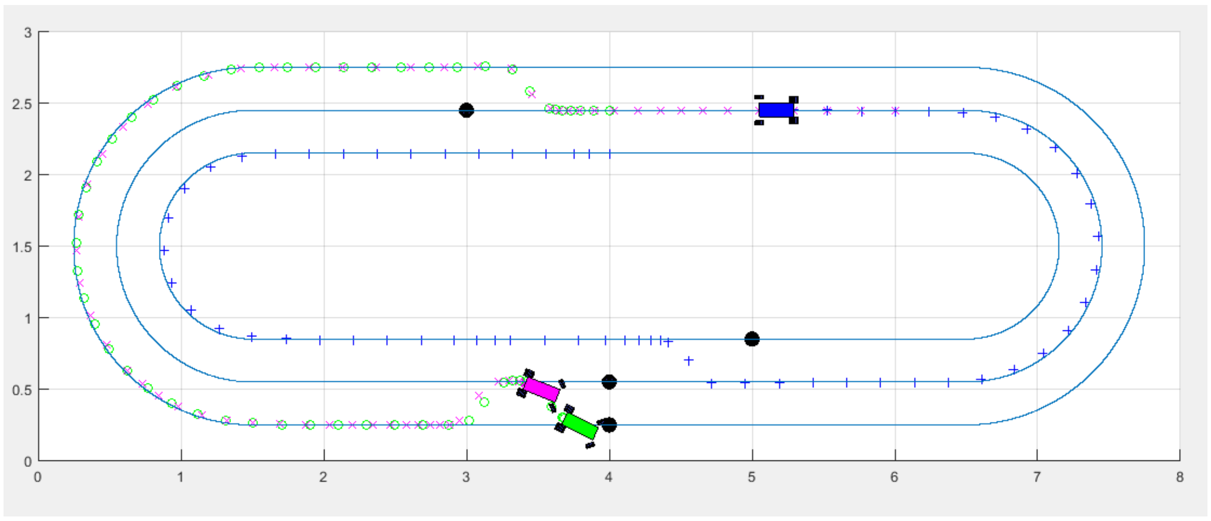


Figure 16 Obstacle avoidance for unavailability of lane [8]

2. The paper proposes four kinds of scheme for obstacle avoidance [11] which are cruising, lane change, decelerating and emergency braking system. The author has used two different models of vehicle, for braking scheme 3DOF model has been used and 2DOF model has been used for cruise and lane change scheme. In lane change scheme the velocity has kept constant, in decelerating scheme, there is no concept lane change. Each scheme is working independently.
3. The idea given in the paper [12] is based on model predictive approach. Lidar sensor is used for obstacle detection. The scheme is presented only for the unstructured environment, and there is no discussion of lane change, as soon as it detects the obstacle, it generates an imaginary boundary around the obstacle and avoids it by keeping safe distance from the imaginary boundary.
4. The approach [13] used the idea of parallax-based information for obstacle detection and avoidance. Parallax is a displacement or difference in the apparent position of an object viewed along two different lines of sight, and is measured by the angle or semi-angle of inclination between those two lines. The parallax angle to obstacles can be defined from the front vertices of the vehicle or from the rear vertices of the vehicle. Using both front and rear parallax angles gives some degree of freedom in weighting the obstacle threat considering its relative position to the vehicle.
5. The approach designed for obstacle avoidance was for the semi-autonomous vehicle [14]. Two scenarios has been designed for the obstacle avoidance. One

scenario is the driver scenario in which driver is attentive and the second scenario is considered where driver is distracted.

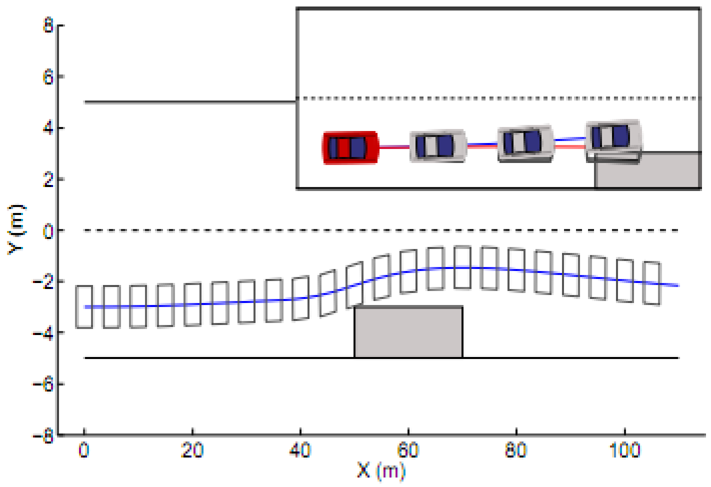


Figure 17 Obstacle avoidance with Driver Assistance [14]

6. Hybrid navigation algorithm, vector field histogram and follow the gap method have also been compared with Impact Point Algorithm and the comparison table is given below in figure 18.

Analysis on Referred Work

The published work was reviewed. The model used in some of the approaches is bicycle model which is two wheel equivalent model of vehicle referred in [8], [9], [10], [11]. An analysis of performance of obstacle avoidance has been reviewed. Each system has achieved its objectives of obstacle avoidance with some assumptions. Some models have used limited obstacles for new velocity and heading angle generation, and some are only applicable in unstructured environment. In unstructured environment there is no concept of lane change because there are no road maps.

Impact Point Algorithm has also achieved the objective of obstacle avoidance and it is based on the predictive modeling of trajectories. It estimated the point of impact and avoids it by changing the lane with required velocity and required heading angle.

Algorithm	Efficiency	Convergence	Time Complexity	Remarks
Vector Field Histogram	Low, calculation may accurate but consumes more memory and processor	No	Requires more time to generate a 2D grid and then conversion to 1D polar histogram	Difficult for micro-controller as high computations are required
Follow the Gap Method	High, always selects shortest path, able to avoid symmetric obstacle	No	Less time consuming as decision are made on the basis of current precepts	Fails in most of the complex cases
Hybrid Navigation	Medium, generates shortest path but no limit to deviate from path	Yes, but in some scenarios car may stop in front of the obstacle	Consume more time in generating reference path	Requires high calculation, consume more time
Impact Point Algorithm	High, generates new heading angle and velocity	Road limit is defined cannot deviate from the path	Does not need much time to calculate	Reference designing makes the cruise control more efficient

Figure 18 Comparison Table of Algorithm

CHAPTER 4 : RESULT ANALYSIS

In order to demonstrate lane changing by vehicle for Obstacle Avoidance, vehicle heading angle is adjusted. A standard lane width of 3.6 metres is considered in this regard

Stationary Obstacle Avoidance

In the first case of obstacle avoidance system a stationary object has been supposed at a finite distance. The controller is calculating the angle of the car with respect to the obstacle position and vehicle speed and its current position. The given below result is the required heading angle of the vehicle. Which increases as soon as the obstacle detected and once the car has started moving according to the required heading angle the angle is decreasing with the time. And once the lane has been changed completely the angle again drops to zero degree.

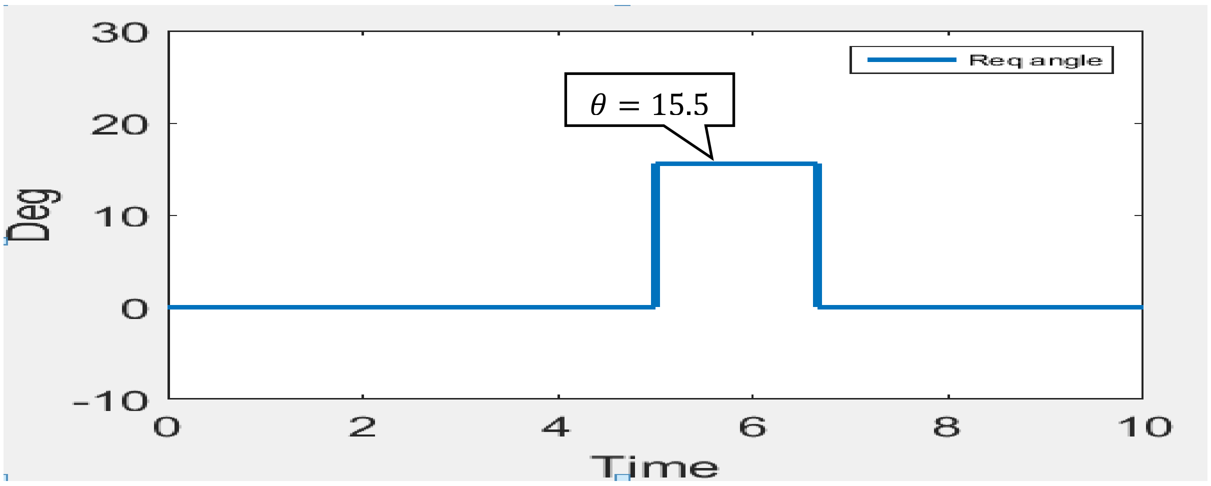


Figure 19 Required Angle to Avoid Stationary Obstacle

In this case the car is moving with velocity of 30km/h, and obstacle was detected after 5 second and to avoid the obstacle the angle calculated is 15.5 degrees. After 5 seconds of the movement of the car the obstacle was detected at the distance of 30m i.e. v*t= 8.33*5=41.67m. So the position of the obstacle is approx. 71.67m. Now, as the no go zone is 2v i.e. 16.66m. So the autonomous car has to reach to second lane within a distance of 13.34m. For that required angle was 15.5 degrees calculated by the impact point algorithm.

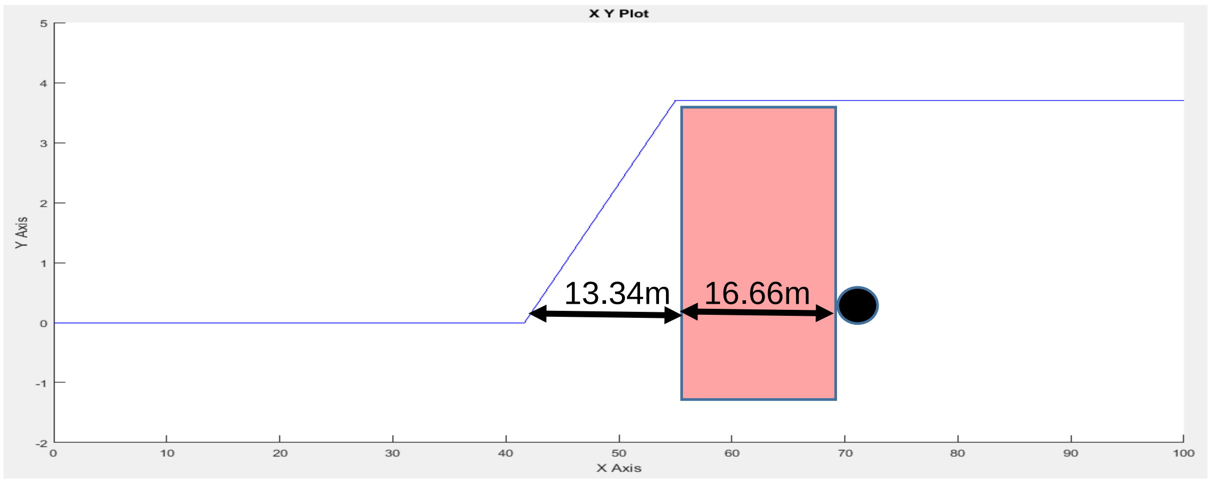


Figure 20 X-Y Graph of Stationary Obstacle Avoidance

Case-2

In case-2 of stationary obstacle, autonomous car has the same velocity as in case-1 i.e. 8.33m/s; and obstacle was detected after 5 sec and it was at a far distance of 100m. To avoid the far object the calculated angle is very small i.e. Angle 2.54 degrees

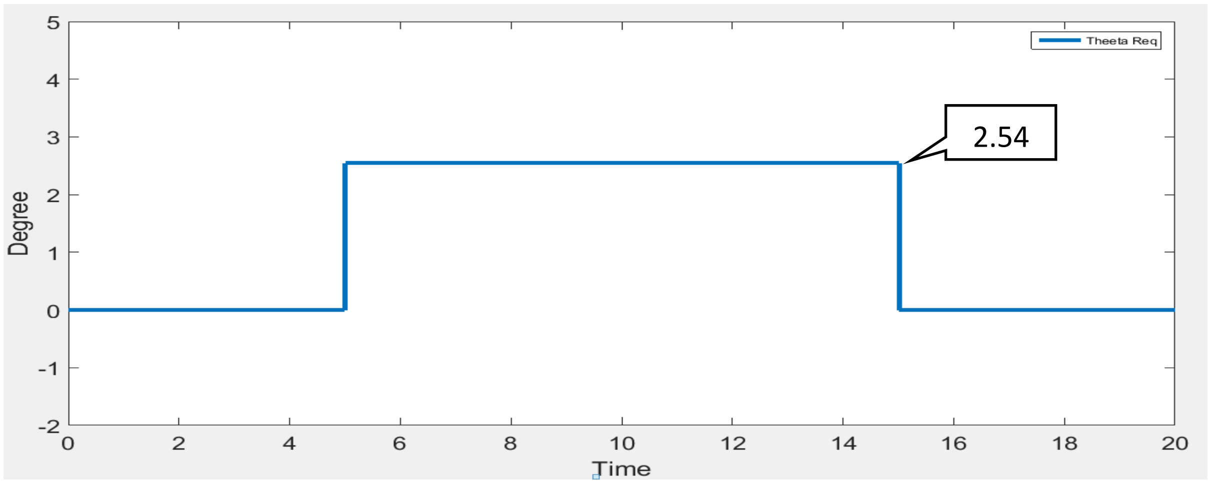


Figure 21 Required Angle for Stationary Obstacle Avoidance

The car was at 41.67m and the obstacle was 100m far, i.e. 141.67m so no go zone is [click] 16.67, so the car has enough distance to change the lane that is why the required angle has been dropped from the 15.5 to 2.54 degrees.

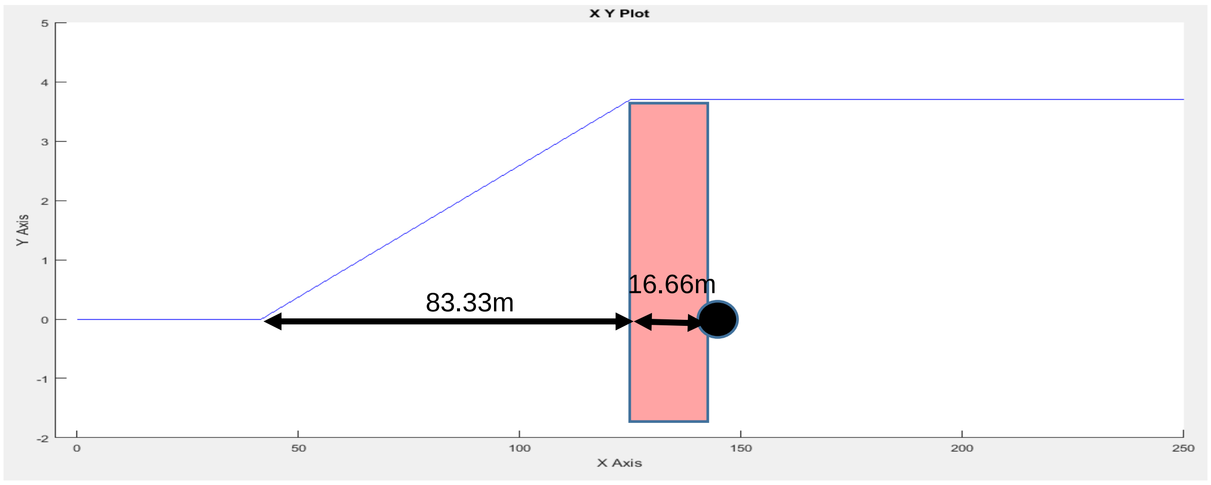


Figure 22 X-Y Graph of Stationary Obstacle Avoidance Case-2

Moving Obstacle Avoidance

For an autonomous car when an obstacle is moving at a certain velocity the required heading angle changes slowly and gradually and when it comes close to the no go zone, the angle increases to avoid the no go zone.

In this particular case, when the obstacle was detected it was at a distance of 130m, the angle generated by the impact point algorithm was 4.1 degrees, but at the same time the distance between both cars was decreasing so the impact point algorithm starts increasing the angle till it reach at the second lane.

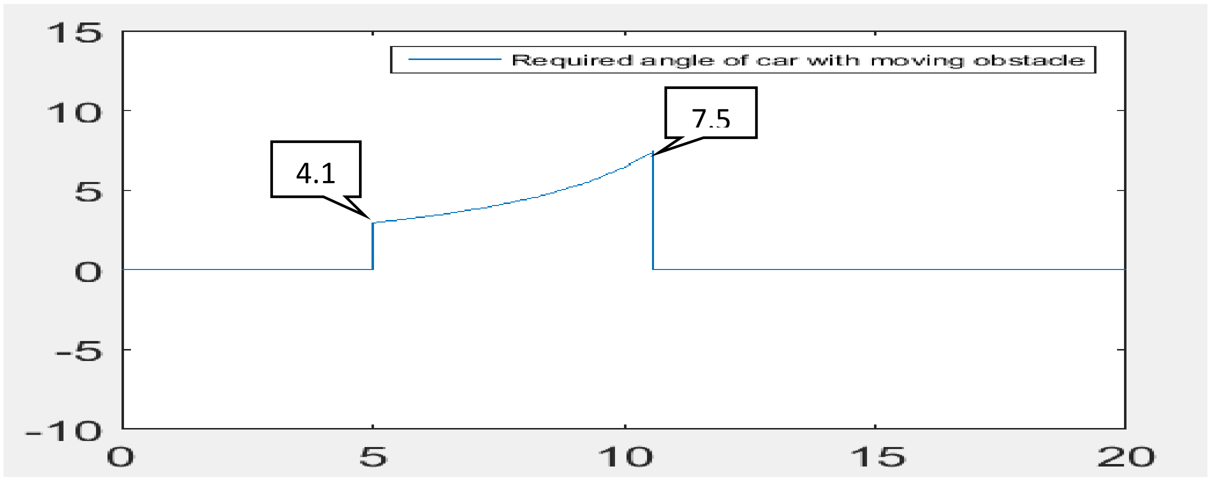


Figure 23 Required Angle for Moving Obstacle Avoidance

Here velocity of the autonomous vehicle is 40km/h i.e. 11.11m/s so after 5 sec it was at the position of 55.56m, velocity of the obstacle is 25km/h, i.e. 6.94m/s after 5 sec it has covered the distance of 34.72m/s so they are coming close at the rate of 4.17m/s.

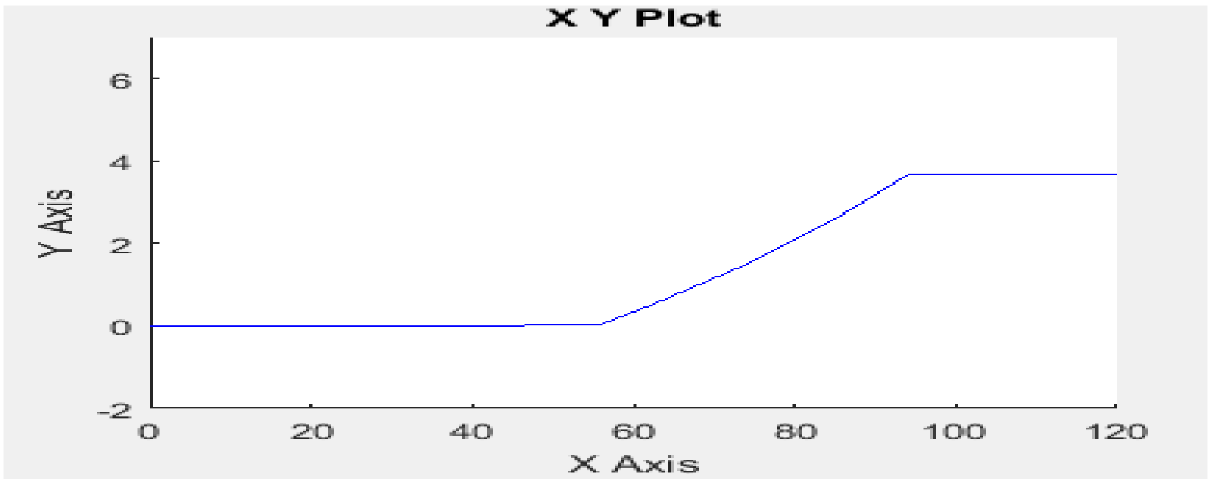


Figure 24 X-Y Graph of Moving Obstacle Avoidance

Case of Velocity Drop

The scenario is avoiding obstacle by decreasing the velocity of the subject vehicle. It also applies in lane change cases when the subject vehicle is approaching the obstacle with higher speed and by just obeying the heading angle command, may result in a fatal accident of the subject vehicle. So it is very much necessary, even for the lane change, that subject vehicle should decrease its speed by certain amount which depends on the obstacle distance, obstacle speed, heading angle, subject vehicle speed and its position.

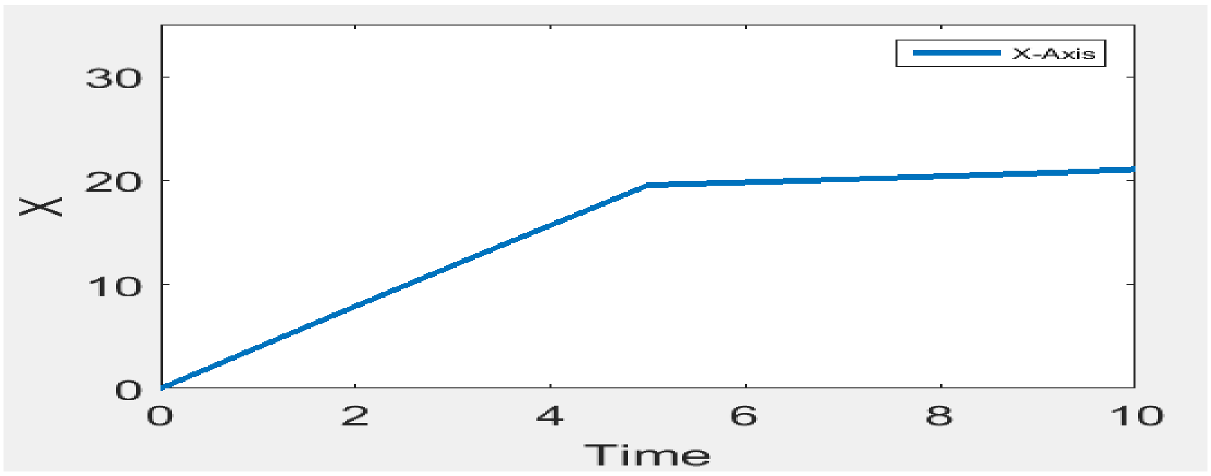


Figure 25 X-Axis Graph of the Velocity Drop Case

The graph of X-axis shows the displacement of the autonomous car with respect to time. In this case, the obstacle was detected at 5sec and both the lanes were occupied so the impact point algorithm has dropped its speed to zero you can see from the above given graph that time is increasing but displacement has stopped.

Now the below graph shows the velocity drop. Autonomous car has assigned the speed of 15km/h i.e. 4.33m/s. At 5 sec the impact point algorithm has generated the command of velocity drop and the velocity has been dropped till 4.33m/s.

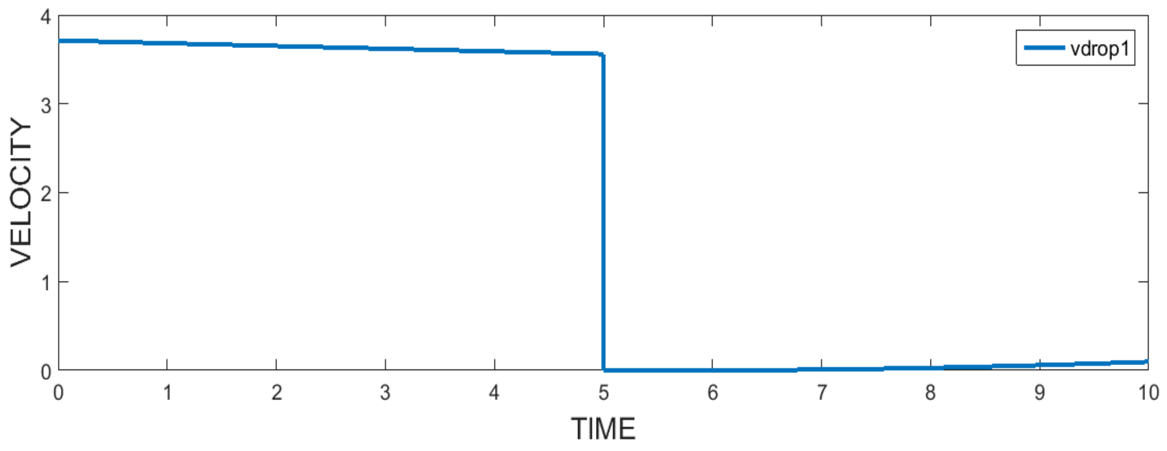


Figure 26 Graph of Velocity Drop

CHAPTER 5 : CONCLUSION

A standalone model of collision avoidance is presented which is capable of avoiding potential obstacles (stationary as well as moving). Different algorithms have been proposed for the detection of objects, prioritization of the obstacle, lane follower and the traffic rule follower are worth mentioning. Mathematical modelling of the Impact Point Algorithm has been derived, which is tested with simulations using a Matlab tool of Simulink®. Obstacles are categorized as stationary and moving because it affect the vehicle heading angle and its speed also. Critical cases have been discussed which are; sensors are not working properly from the very far distance and obstacle has been detected at a finite distance. These situations need immediate line of action to avoid the obstacle which may result in fatal accidents. In the last but not least, the results are being analysed and explained in a separate chapter.

CHAPTER 6 : FUTURE WORK

Security Checks

Some obstacle are those which are no threats to the vehicle and does not cause any harm, and some are small in size but can create a lot of mess. Obstacles may be divided into different categories according to the threat they possess.

1.1. Low/No Risk Obstacle

1.2. Low to Medium Risk Obstacle

1.3. Medium to High Risk Obstacle

1.4. High Risk Obstacle

1.5. Critical Obstacle

This is important because there are times when the vehicle has to decide between different obstacles.

Road Class

Road class decides the velocity of the vehicle and the speed limit on that road. Road class may add, it contains the highway road, primary and secondary roads of the town. There are also some roads which are categorized as local/urban class and vehicle may have to run on the trails as well. Each type contains different rules and different speed limits.

Road Surface/Condition

The condition of the road also affects the performance and cost function of the control. One may include the road surface in its model for high accuracy.

3.1.	Compacted (Rough)	<40kph
3.2.	Compacted (Smooth)	>40kph
3.3.	UN-Compacted (Rough)	<40kph
3.4.	UN-Compacted (Smooth)	<40kph
3.5.	Snow/ Ice	

3.6. Mud/Sand

Obstacles

More Obstacle can be added like;

4.1. Pedestrians

4.2. Check Points

4.3. Debris/Wreckage

4.4. Downed Electric Lines

4.5. Landslide/ Mudslide

4.6. Wet crossing

4.7. Road Damage

4.8. Bridge

4.9. Cautionary Signs on the road

4.10. Elevation

4.11. Restricted Area

4.12. Traffic Lights

Last but not the least, the driverless vehicle should have the capability to prioritize the potential threat and take the decision accordingly.

Implementation

A practical implementation of this work may be carried out. The designed algorithm be implemented on any Vehicle to make it autonomous. A set of electronics would be required to see the hardware implementation. In the same chain a real time hardware-in-loop simulation can be prepared where a controller on board may be prepared and implemented in the vehicle.

REFERENCES

[1] R. Whitwam, "How Google's Self-driving cars detect and avoid obstacles," in *www.extremetech.com*, 2014.

[2] P. E. Ross, "Tesla's Model S will offer 360-degree Sonar," in *IEEE Spectrum*, 2014.

[3] V. Sezer, "A Novel Obstacle Avoidance Algorithm: "Follow the Gap Method"," *Elsevier,* 2012.

[4] M. Zohaib, "Control Strategies for Mobile Robot with Obstacle Avoidance," Islamabad.

[5] B. I. J Oroko, "Obstacle Avoidance and PAth Planning Schemes for Autonomous Navigation of a Mobile Robot," in *Sustainable Research and Innovation Proceedings*.

[6] T. Z. J. S. X. L. Y Zhu, "A New Hybrid Naviagtion Algorithm for Mobile Robots in Environments with Incomplete Knowledge," in *Knowledge Based Systems*, 2012.

[7] DARPA, DARPA Grand Challenge website, United States Department of Defence [http://www.darpa.mil/grandchallenge/], 2008.

[8] W. K. Grefe, Integrating Collision Avoidance, Lane Keeping, and Cruise Controlwith an Optimal Controller and Fuzzy Controller, Virginia: Virginia Polytechnic Institute and State University, 2005.

[9] David A. Dickey, N Carolina State U, Raleigh, Introduction to Predictive Modelling with Examples, SAS Global Forum, 2012.

[10] T. Shrivastava, Perfect way to build a Predictive Model, https://www.analyticsvidhya.com, 2015.

[11] Hongyan Guo, Rui Jia, Zaitao Yu, Obstacle Avoidance for Autonomous Ground Vehicles based on Moving Horizon Optimazation, Shenyang, China: Preceeding of the 11th world congress on Intelligent Control and Automation, 2014.

[12] Jiechao Liu, Paramsothy Jayakumar, An MPC Algorithm with Combined Speed and Steering Control for Obstacle avoidance in AGV, Ohio, USA: Preceedings of ASME 2015 Dynamic Systems and Control Conference, 2015.

[13] J M Park, D W Kim, Y S Yoon, H J Kim, K S Yi, Obstacle Avoidance of Autonomous Vehicles based on Model Predictive Control, Seoul, Korea, 2009.

[14] Andrew Gray, Muhammad Ali, Yiqi Gao, Semi-Autonomous Vehicle Control for Road Departure and Obstacle Avoidance, University of Califorinia, Berkeley, USA.

[15] A. K. J Anupama, "Design and Development of Autonomous Ground Vehicle for Wild Life Monitoring," *International Journal of Innovative Research in Computer and Communication Engineering,* vol. 2, no. 5, 2014.

[16] D. Langer, "Autonomous Driving and Intelligent Vehicles," in *Volkswagan Electronics Research Laboratory*, 2012.

[17] K. You, "Autonomous NAvigation and Obstacle Avoidance Vehicle," Florida, 2008.

[18] C. L. Kumari, "Building Algorithm for Obstacle Detection and Avoidance System for Wheeled Mobile Robot," in *Global Journal of Researches in Engineering Electrical and Electronics Engineering*, 2012.

[19] N. P. David Vissiere, Experiments of trajectory generation and obstacle avoidance for UGV, The American Control Conference, IEEEXplore, 2007.

[20] S. S. J. E. M. &. R. O. S. D E Chang, Collision Avoidance for Multiple Agent System, 42nd IEEE Conference of Decision and Control, 2003.

[21] A. V. &. AzimEskandarian, Research advances in Intelligent Collision Avoidance and Adaptive Cruise Control, IEEE Transactions on Intelligent Transportation Systems, 2003.

[22] G. Li, "An Efficient Improved Artificial Potential Field based Regression Search Method for Robot Path Planning," in *Internation Conference on Mechatronics and Automation (ICMA), IEEE*, 2012.

[23] I. Kamon, A Range sensor based Navigation Algorithm, The International Journal of Robotics Research, 1998.

[24] S. K. Kalmegh, "Obstacle Avoidance for a Mobile Exploration Robot using a Single Ultrasonic Range Sensor," in *Emerging Trends in Robotics and Communication Technologies (INTERACT), IEEE*, 2010.

[25] V.Sezer, "A New Fuzzy Speed Control Strategy Considering Lateral Vehicle Dynamics," in *IEEE Conference on Intelligent Transportation System*, Alaska, USA, 2012.

APPENDIX A

SIMULINK MODELS

Stationary Obstacle Avoidance

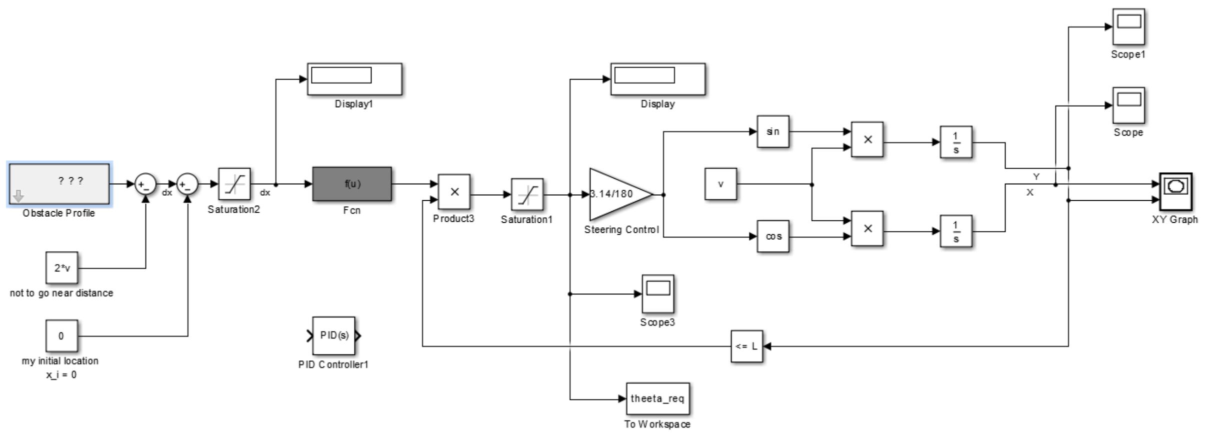


Figure 27 Stationary Obstacle Avoidance Simulink Model

Moving Obstacle Avoidance

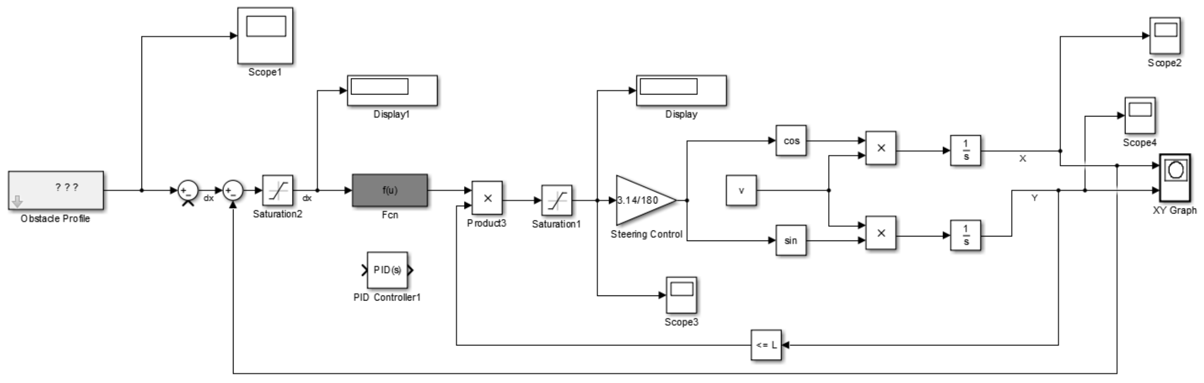


Figure 28 Moving Obstacle Avoidance Simulink Model

Velocity Drop Model

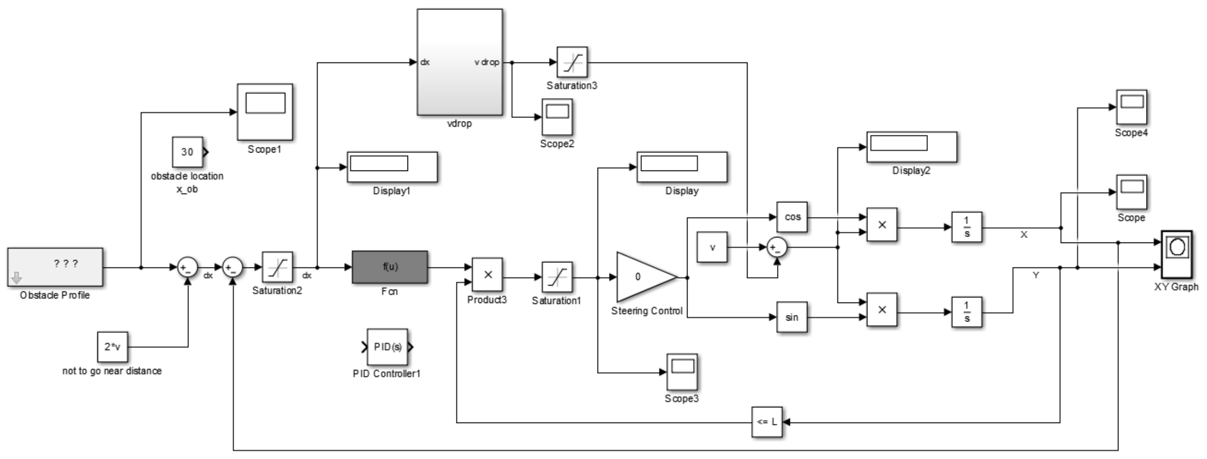


Figure 29 Velocity Drop Simulink Model

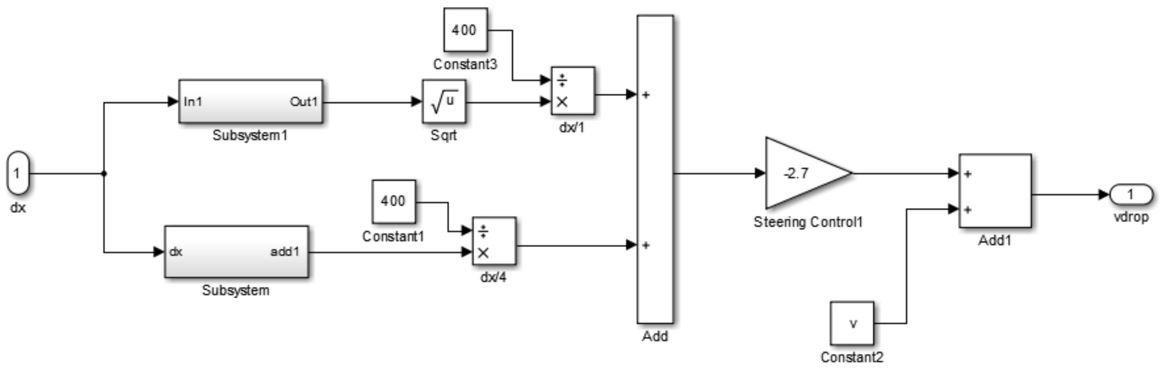


Figure 30 Velocity Drop Box

APPENDIX B

CODES

Graphical User Interface of Impact Point Algorithm

```
function varargout = OA_Software_v1(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @OA_Software_v1_OpeningFcn, ...
                   'gui_OutputFcn',  @OA_Software_v1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before OA_Software_v1 is made visible.
function OA_Software_v1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to OA_Software_v1 (see VARARGIN)
global h
clc;
h.veh.v=60/3.6;
h.veh.x=0;
h.lanewidth=3.6;
h.veh.y= (2 - 1) *h.lanewidth;
h.obs.v=0;
h.obs.x=100;
h.obs.y=h.lanewidth;
h.laneavailability=1;
% h.Euler=[20;0;0]*3.14/180;
% h.f=[500;500;500;500];
% h.m=1430;
% h.g=9.81;
h.simtime=1000/100;    %in sec
% Choose default command line output for OA_Software_v1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes OA_Software_v1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = OA_Software_v1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function roll_Callback(hObject, eventdata, handles)
% hObject    handle to roll (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of roll as text
%        str2double(get(hObject,'String')) returns contents of roll as a
double
global h
h.veh.v=str2double(get(hObject,'String'))/3.6;

% --- Executes during object creation, after setting all properties.
function roll_CreateFcn(hObject, eventdata, handles)
% hObject    handle to roll (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pitch_Callback(hObject, eventdata, handles)
% hObject    handle to pitch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pitch as text
%        str2double(get(hObject,'String')) returns contents of pitch as a
double
global h
h.veh.x=str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function pitch_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pitch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

function yaw_Callback(hObject, eventdata, handles)
% hObject    handle to yaw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of yaw as text
%        str2double(get(hObject,'String')) returns contents of yaw as a
double
global h
h.veh.y=  (str2double(get(hObject,'String'))-1) *3.6;

% --- Executes during object creation, after setting all properties.
function yaw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to yaw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double
global h
h.obs.v=str2double(get(hObject,'String'))/3.6;

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double
global h
```

```
h.obs.x=str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double
global h
h.obs.y=(str2double(get(hObject,'String'))-1) *3.6;

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a
double
global h
h.f(4)=str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double
global h
h.m=str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a
double
global h
h.simtime=str2double(get(hObject,'String'))/100;

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global h

if h.obs.v~=0
    if h.laneavailability==1
        sim('one_moving');
        figure('outerposition',[100 10 1200 600],'title','Obstacle
Avoidance Software')
        subplot(1,2,1);
        plot(theeta_req.Time, theeta_req.Data,'LineWidth',3)
        axis([0 max(theeta_req.Time) min(theeta_req.Data)-2
max(theeta_req.Data)]+2);
        set(gca,'fontsize',14,'LineWidth',2);
        xlabel('Time(t) in sec','fontsize',16);
        ylabel('Req Heading Angle in deg','fontsize',16);
%     legend('Desired heading','Current heading')
        subplot(1,2,2);
        plot(x.Data,y.Data,'LineWidth',3)
        set(gca,'fontsize',14,'LineWidth',2);
    axis([-1 max(x.Data) -1.8 16.2])
        xlabel('Position (x) in m','fontsize',16);
        ylabel('Position (y) in m','fontsize',16);

    else
        one_moving_vdrop;
    end
else
    sim('one_stationary');
    figure('outerposition',[100 100 1200 600])
    subplot(1,2,1);
    plot(theeta_req.Time, theeta_req.Data,'LineWidth',3)
    axis([0 max(theeta_req.Time) min(theeta_req.Data)-2
max(theeta_req.Data)]+2);
    set(gca,'fontsize',14,'LineWidth',2);
    xlabel('Time(t) in sec','fontsize',16);
    ylabel('Req Heading Angle in deg','fontsize',16);
%     legend('Desired heading','Current heading')
    subplot(1,2,2);
    plot(x.Data,y.Data,'LineWidth',3)
    set(gca,'fontsize',14,'LineWidth',2);
    axis([-1 max(x.Data) -1.8 16.2])
    xlabel('Position (x) in m','fontsize',16);
    ylabel('Position (y) in m','fontsize',16);

end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all

% --- Executes during object creation, after setting all properties.
function ipa_CreateFcn(hObject, eventdata, handles)
axes(hObject)
image(imread(strcat('impact point algorithm','.png')));
axis off
```