Sensor Fusion and Non-linear MPC controller development studies for Intelligent Autonomous vehicular systems

Ahammad Basha Dudekula
Michigan Technological University, adudekul@mtu.edu

SENSOR FUSION AND NON-LINEAR MPC CONTROLLER DEVELOPMENT STUDIES FOR INTELLIGENT AUTONOMOUS VEHICULAR SYSTEMS

By
Ahammad Basha Dudekula

A DISSERTATION
Submitted in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
In Mechanical Engineering-Engineering Mechanics

MICHIGAN TECHNOLOGICAL UNIVERSITY
2020

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Mechanical Engineering-Engineering Mechanics.

Department of Mechanical Engineering-Engineering Mechanics

Dissertation Advisor:	*Dr. Jeffrey D. Naber*
Committee Member:	*Dr. Bo Chen*
Committee Member:	*Dr. Jeremy Worm*
Committee Member:	*Dr. Stephen A. Hackney*
Department Chair:	*Dr. William W. Predebon*

Dedication

To my Parents, Wife, Guide and Brother

who have provided continuous support and without which this work would have not been achieved today.

Contents

List of Figures

List of Tables

Author Contribution Statement

The Author contributed in several projects, in which some of the projects published into journals and some of them are under review. The list is furnished below as follows:

† Chapter.2 is accepted for publication in SAE 2020, and I am first author for the publication. The work focuses on sensor fusion using Modified Extended Kalman Filter for obtaining vehicle heading for the autonomous vehicle application. The experiments were conducted at APSRC using $1/5^{th}$ truck and Python has been used for real-time implementation and data analysis.Dr.Naber has proofread the manuscript and stand as co-author for the publication.

† Chapter.3 is submitted in SAE Connected and Automation Vehicles and I am first author for the publication. The work mainly focuses on developing an algorithm using Non linear MPC controller for avoiding road grade, stationary and moving obstacle in an unstructured environment. The MATLAB with CaSAi tool has been used to develop the algorithm. Further, a novel obstacle-grade avoidance method has been proposed in the work and the manuscript has been proofread by Dr.Naber and acts as co-author for the publication.

† Chapter.4 is under review to publish in IEEE intelligent systems and I am first author for the publication. The publication focuses on simulation and implementation of LQG algorithm on $1/5^{th}$ truck for path tracking while assessing considerable sensor and environmental disturbance on the path. Python tool has been used for developing LQG simulations and implementing it on real vehicle. The Beagle Bone Black Embedded system has been used for real-time implementation of LQG algorithm on vehicle and it interfaces with various sensors including GPS base station, IMU, Speed sensor and steering measurement module. Further, the sensor fusion algorithm using Kalman filter for making vehicle state estimates to reject the environmental disturbances have been developed. The desired path has been made using Cubic spline fit. The manuscript is under review and Dr.Naber proofreads the document and acts as co-author for the publication.

† The Authour has contributed to several other projects including $1/5^{th}$ car assembly, real time data plotting for NEXTCAR project, Leddar VU8 sensor studies for Volt vehicle in ARPA-e project, Volt Gen-1 model development with Fuzzy logic control in association with Hitachi company.

Acknowledgments

I would like to thank Michigan Technological University and ARPA-e team for providing funding for the research work. Acknowledgements are provided to Graduate school and mechanical department for providing continuous support for my doctoral candidacy.
Though it seems simple to express my gratitude towards my guide Dr.Jeffrey D Naber, he has been my sole inspiration to complete PhD studies. The journey started in Spring 2015 and he has been instrumental in proving valuable technical suggestions including purchasing autonomous vehicle, GPS base station, heading estimation modules. The technical discussions of controller development, vehicle model developments and sensor fusion studies helped me to formulate the problem in systematic manner. The sensors were getting failed and experiments were failed numerous times during the research work. However, my advisor was keep encouraging me to have faith and inspired me at every stage of my failure. The most important attributes, I learned from my advisor is to have dedication, hard work and discipline towards the research work. I thoroughly enjoyed the freedom he has given to me and I certainly owe a big thanks to him. It is always been my pleasure to work with him and I am fortunate to have such a good mentor in my life.
I am very grateful to have Dr. Bo Chen, a passionate and knowledgeable person in my committee, since Summer 2016. She has provided valuable suggestions in the proposal defense and acts as active member in ARPA-e projects. These projects given me the opportunity to explore different control algorithms.
I owe many thanks to Dr.Jeremy for supporting my research and providing lab space for the research work. I am fortunate to have such hands on experience person in my advising committee.
I would also like to thank Dr. Stephen Hackney for serving as my PhD committee member and his valuable lessons in Battery design helped me to chose the appropriate vehicle for the research work.
I would like to mention special thanks to Paul, who have given many suggestions during the building of 1/5th truck and for allotting space and instruments in the Electronics lab. The GPS base station require near window operation and Paul has allotted space and encouraged me for conducting the research in APS labs.
I would also like to thank Master students Shuvodeep, Shubham, Raghu for helping me to build the 1/5th truck. The project was started from scratch and Shuvodeep helped in procuring the hardware and helped me to setting up the sensor interface. There were many situations, where we worked whole night for debugging the wiring issues. I would really appreciate Shubham for designing and modifying the vehicle body and chassis for conducting the tests even during wintry conditions. The enclosed design allowed me to arrange controller, sensors and external batteries in the

vehicle and can operate during the houghton winter conditions.
I would like to thank my best friends Gopi, Ramana, Abdulla and Nari for their support during my early career growth and it is the right time to convey my gratitude towards them. I learned many things from them and I would certainly cherish their memories at every stage of my life. I would also like to thank, my role model and Master's advisor Prof. Venkateshan, from IIT Madras, for his encouragement, care and unconditional faith on me.
I am also very fortunate to have some of the best colleagues, who made my research enjoyable at APS labs. Stats,Shreyas,Niranjan,Behrouz and Zhou are some of the fellow PhD students had great discussions and provided valuable suggestions during the tough times. I would also like to give my appreciation to Master students Gaurav,Vishal,Mahesh and Vasu, who have worked with me provided support in handling research material and weekly meetings. I would like to thank all APS crew for providing support for conducting the research work. Special thanks to my friends at Michigan tech university including Zakkam, Yaswanth,Siva, Laxmi, Moiz, Rajesh, Hemanth, Sumanth, Sashank,Nitin,Pathak and many others.
Last but not least, special thanks to my family, without their support, I wouldn't have imagined the PhD degree. I blessed to have supportive parents, Husenamma, Pedda Dastagiri Garu, who have always encouraged me to chose right path and given freedom to go beyond the limits. My Wife, Hussain Bee, who have all the patience in world and cared me at every stage of my life. I owe her a big Thank you for her continuous support and understanding me at tough times.My brother, Mahaboob basha, who have vital role in my life and had continuous, unconditional faith on my abilities. I am really fortunate to have such family members and finally my two lovely kids Aryan and Arshad, with whom I would love to play and forget the rest of world.

Definitions

Levels of Autonomy:

The Society of Automotive Engineering (*SAE*) has defined total of 6 levels from fully manual mode to fully autonomous mode.

Level-0: Manual mode

There is no driving automation during this mode and it is simply the current conventional vehicle operation. Though, some features including emergency braking would help driver, humans provide *dynamic driving task* and does not qualify for the automation.

Level-1: Driver Assistance mode

The vehicle equipped with single automated feature of either steering or acceleration of the vehicle (cruise control). This adaptive cruise control assist driver in maintaining the safe distance from next car.

Level-2: Partial Automation mode

This mode is also called Advanced Drive Assistance system or ADAS. Here, the vehicle can control steering as well as acceleration and deceleration. However, the Driver still has to sits in the driver take control of car at any time.

Level-3: Conditional Automation mode

Level-3 is a critical transformation for autonomous vehicle development. These vehicles have ability to detect surroundings and can make informed decision for themselves such as overtaking on slow moving vehicle. However, driver must remain alert and ready to take control of it, when the controller could not able to execute the task.

Level-4: High Automation mode

This mode of vehicle dose not require human intervention in most circumstances. The main difference between Level and level- 4 is that, Level-4 vehicle can intervene and self correct, if things go wrong or system failure occurs. However, human still have option of overtaking the vehicle in manual mode.

Level-5: Full Automation mode

These vehicle do not require human attention and the human *dynamic driving task* is eliminated. These vehicles would not even have steering or acceleration/braking pedals.

List of Abbreviations

(x_g, y_g)	Target position, $[m]$.
(x_0, y_0)	Vehicle current position w.r.to vehicle C.G. location, $[m]$.
(x_{om}, y_{om})	measured obstacle position w.r.to vehicle current position, $[m]$.
(x_{op}, y_{op})	Predicted obstacle position w.r.to vehicle current position, $[m]$.
$(x^{(j)}_{mobs}, y^{(j)}_{mobs})$	j^{th} moving obstacle position w.r.to vehicle current position, $[m]$.
$v^{(j)}_{mobs}$ j^{th}	moving obstacle longitudinal velocity, m/s.
$\psi^{(j)}_{mobs}$	j^{th} moving obstacle angle of shoot, $[°]$.
$O^{(j)}_{m,i}$	j^{th} measured obstacle position, with $i = 0, 1, ..n$ predicted positions,where $j = 1, 2, 3, ..k$ detected obstacles in LIDAR view with $O^{(1)}_{m,0} = (x^{(1)}_{om,0}, y^{(1)}_{om,0}), O^{(2)}_{m,1} = (x^{(2)}_{om,1}, y^{(2)}_{om,1})$.., in $[m]$.
P_i	vehicle predicted Positions, where $i = 0, 1, 2, ..N$ predictions with $P_0 = (x_0, y_0), P_1 = (x_1, y_1)$.., in $[m]$.
$P_{mv,i}$	vehicle predicted Positions for moving obstacle analysis, where $i = 0, 1, 2, ..n$ predictions with $P_{mv,0} = (x_{mv,0}, y_{mv,0}), P_{m,1} = (x_{mv,1}, y_{mv,1})$.., in $[m]$.
α_f, α_r	Front and rear slip angle in $[rad]$.
ψ_{mobs}	Angle of shoot or heading of moving obstacle in $[rad]$.
ψ_{ulimit}	Heading upper limit for dangerous moving obstacle in $[rad]$.
ψ_{llimit}	Heading lower limit for dangerous moving obstacle in $[rad]$.
$\dot{\psi}$	vehicle yaw rate in $[rad/sec]$.
γ_f	vehicle steering rate in $[rad/s]$.
θ	Road grade in $[rad]$.
$\mu_{z,x}$	vehicle longitudinal load transfer coefficient in $[N/(m/s^2)]$.
$\mu_{z,yf}, \mu_{z,yr}$	vehicle lateral load transfer coefficients in front,rear axle $[N/(m/s^2)]$.
ψ	Vehicle heading or yaw in $[rad]$.
a, b	penalty terms in cost function for vertical load, , $[N]$.
a_x	vehicle longitudinal acceleration in $[m/s^2]$.
d_0	Distance between vehicle current position and Target location in $[m]$.
d_f	Distance between vehicle end of prediction position and Target location in $[m]$.
$D_{s_{mobs}}$	vehicle lateral speed in $[m/s]$.
$D_{v_{mobs}}$	vehicle lateral speed in $[m/s]$.
F_{grade}	Grade force in $[m/s]$.
F_{yf}, F_{yr}	Lateral force on front,rear axle in $[N]$.
$F_{z,f}, F_{z,r}$	Dynamic vertical load on front,rear axle in $[N]$.
$F_{z,f0}, F_{z,r0}$	Static vertical load on front,rear axle in $[N]$.

$F_{z_{thr}}$	Tire threshold vertical load in $[N]$.
$F_{z_{off}}$	vehicle lateral speed in $[m/s]$.
I_z	moment of inertial about z-axis in $[kg - m^2]$.
J_f	vehicle Jerk in $[m/s^3]$.
l_f, l_r	Distance from C.G. location to front/rear axle $[m]$.
l	Wheel base in $[m]$, $(l_f + l_r)$.
L_d	LIDAR sensor detection range, in $[m]$.
L_p	Prediction horizon length/distance in $[m]$.
M_{u_f}, m_{u_r}	Unsprung mass of front,rear axle side in $[kg]$.
M_s	Sprung mass in $[Kg]$.
M_{total}	Total mass in $[Kg]$.
N	Number of predictions.
O_d	Distance between current vehicle position to stationary obstacle position in $[m]$.
O_{mobs}	Distance between current vehicle position to moving obstacle position in $[m]$.
St_{thr}	vehicle lateral speed in $[m/s]$.
T_e	MPC Execution horizon in $[sec]$.
T_p	vehicle prediction horizon in $[s]$.
T_p	Prediction horizon in $[sec]$.
v_{mobs}	Velocity of moving obstacle in $[m/s]$.
v_x	vehicle longitudinal speed in $[m/s]$.
v_y	vehicle lateral speed in $[m/s]$.
$\mathcal{U}$	Control vector.
$\mathcal{Z}$	State vector.
$\mathcal{D}(\cdot)$	Generic vehicle dynamic model function.
$\mathcal{L}(\cdot)$	Generic prediction length constraint.
$\mathcal{O}_{st}(\cdot)$	Generic stationary obstacle avoidance constraint.
$\mathcal{O}_{mv}(\cdot)$	Generic moving obstacle avoidance constraint.
$\mathcal{S}(\cdot)$	Generic dynamical safety constraint.
$\mathcal{T}(\cdot)$	Generic terminal cost.
$\mathcal{F}(\cdot)$	Pacejka magic tire model function.

Abstract

The demand for safety and fuel efficiency on ground vehicles and advancement in embedded systems created the opportunity to develop Autonomous controller. The present thesis work is three fold and it encompasses all elements that are required to prototype the autonomous intelligent system including simulation, state handling and real time implementation. The Autonomous vehicle operation is mainly dependent upon accurate state estimation and thus a major concern of implementing the autonomous navigation is obtaining robust and accurate data from sensors. This is especially true, in case of Inertial Measurement Unit (IMU) sensor data. The IMU consists of a 3-axis gyro, 3-axis accelerometer, and 3-axis magnetometer. The IMU provides vehicle orientation in 3D space in terms of yaw, roll and pitch. Out of which, yaw is a major parameter to control the ground vehicle's lateral position during navigation. The accelerometer is responsible for attitude (roll-pitch) estimates and magnetometer is responsible for yaw estimates. However, the magnetometer is prone to environmental magnetic disturbances which induce errors in the measurement. The initial work focuses on alleviating magnetic disturbances for ground vehicles by fusing the vehicle kinematics information with IMU senor in an Extended Kalman filter (EKF) with the vehicle orientation represented using Quaternions.

The previous studies covers the handling of sensor noise data for vehicle yaw estimations and the same approach can be applied for additional sensors used in the work. However, it is important to develop simulations to analyze the autonomous navigation for various road, obstacles and grade conditions. These simulations serve base platform for real time implementation and provide the opportunity to implement it on real road vehicular application and leads to prototype the controller. Therefore, the next section deals with simulations that focuses on developing Non-linear Model Predictive controller for high speed off-road autonomous vehicle, which avoids undesirable conditions including stationary obstacles, moving obstacles and steep regions while maintaining the vehicle safety from rollover. The NMPC controller is developed using CasADi tools in MATLAB environment.

As mentioned, the above two sections provide base platform for real time implementation and the final section uses these techniques for developing intelligent autonomous vehicular system that would track the given path and avoid static obstacles by rejecting the considerable environmental disturbance in the given path. The Linear Quadratic Gaussian (LQG) is developed for the present application, The model developed in the LQG controller is a kinematic bicycle model, that mimics $1/5^{th}$ scale truck and cubic spline has been used to connect and generate the continuous target path.

Chapter 1

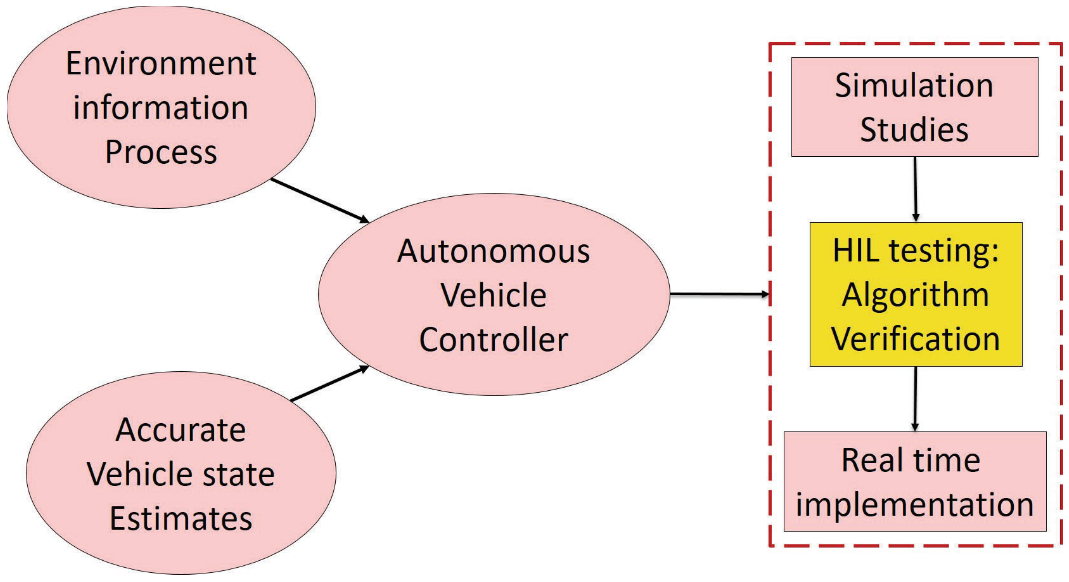
Introduction

1.1 Background and Motivation

The technology advancements in embedded processing, GPS accuracy and efficient control algorithms provide opportunity to develop full fledged autonomous vehicle to meet safety and emissions. The future predictions from ***Victoria Transportation Policy Institute*** indicated that the vehicle manufactures tend to produce half of new vehicles are autonomous by 2045 and half of the vehicle fleet is autonomous by 2060 [1]. Further, the unmanned buses and delivery trucks would become common by 2030 and optimistically the commercial cars may be estimated to be available for the safe and reliable operation by 2030. It is further predicted that, by 2060, there is 40% increased safety, 35% improvement in congestion and emissions and 30% improvement in driver stress fatalities. However, the predictions are made with optimistic perspective and there were many past experiences with vehicle crashes on both on road and off road applications.

1.1.1 Critical Areas in Autonomous Vehicle Development

The major challenges in developing the autonomous controller is to obtain robust and noise free sensor data, efficient algorithm development for fast processing on embedded systems and simulations and real time implementation of developed control algorithms on actual vehicle is shown in Fig.1.1. The first challenge require development of several fusion algorithms for processing sensor information and making best



Environment information Process
Accurate Vehicle state Estimates
Autonomous Vehicle Controller
Simulation Studies
HIL testing: Algorithm Verification
Real time implementation

Figure 1.1: Critical areas for developing autonomous vehicle.

estimates for obtaining vehicle surrounding information. For instance, the vehicle yaw or heading is critical parameter in developing autonomous navigation and it is prone to external magnetic fields. Therefore, it is important to obtain robust and correct heading data, when vehicle is affected with unwanted, sudden magnetic fields on the path. The present work deals this problem by developing the modified extended kalman filter for fusing the vehicle kinematics with IMU magnetometer data. Further, this algorithm has been developed using Python script and used in the real time implementation of LQG controller on actual vehicle. The developed algorithm has improved the LQG controller performance and eliminated the unwanted external magnetic field effects during the course of desired path tracking on $1/5^{th}$ truck.
The second challenge is to develop efficient control algorithms that should not only provide robust and safety operation but also should meet the implementation requirements. However, to ensure robustness and safety in control algorithms, proper methodology for handling vehicle dynamics model and constraints are required. The inaccuracy in dynamic model and constraint handling leads fatal vehicle crash. Further, it is important to verify the algorithm in simulations for saving the cost and time. The Model predictive controller algorithm best suits for present application, as it foresees the future in advance and reacts to the current situation. However, the inaccuracy in model development and uncertainty in sensor information can lead to vehicle crash. The careful consideration of vehicle dynamics, vehicle rollover safety and other constraints requires thorough study on vehicle lateral and longitudinal dynamics and obstacle avoidance algorithms. This challenge has been undertaken in the current research work and developed a non-linear MPC algorithm that would avoid both moving and stationary obstacles for high speed off road autonomous vehicles. These vehicles can carry human needs to the destination in the off-road conditions,

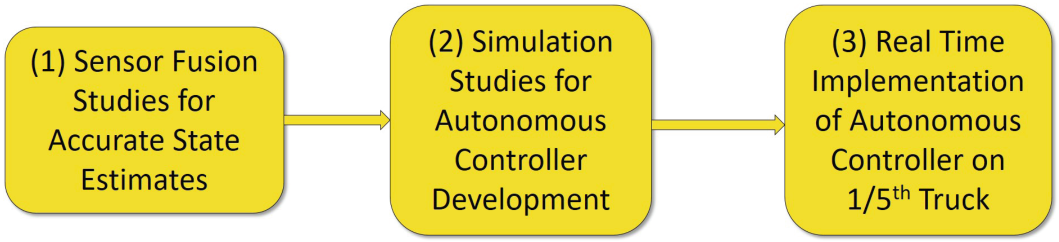

(1) Sensor Fusion Studies for Accurate State Estimates
(2) Simulation Studies for Autonomous Controller Development
(3) Real Time Implementation of Autonomous Controller on 1/5th Truck

Figure 1.2: Overview of the present research work.

while maintaining the safety from both moving and stationary obstacle. The military sections, forest or unstructured path rescue operations can best utilize the features of current research work.
The third challenge in developing the autonomous vehicle is to implement the developed control algorithms on actual vehicle. Though, the development of non-linear MPC controller for both moving and stationary obstacles for high speed application are made through simulations, the real-time implementation of control algorithm on actual vehicle is still implausible due to computational burden and vehicle integration. The computational burden can be solved by using the fast processing embedded systems and off-loading some of the calculations including terrain map processing,moving obstacle processing etc through work stations. On the otherhand, the vehicle integration adds up additional effort in debugging the existing CAN signals for autonomous controller. Therefore, in the current research work, the real time implementation has been done on $1/5^{th}$ Electric truck with the LQG controller algorithm. The work has laid out the perfect platform for future MPC controller implementations and debugging of vehicle motors and actuator signals have been made. Further, it is easy to export the developed algorithms into actual vehicle as it has similar powertrain and actuators for the autonomous navigation. This save lot of time,effort and cost. However, the present work does not included the Hardware In Loop (HIL) tests and it is left for the future work.

1.1.2 Putting pieces together

The present work is three fold including sensor fusion, controller development using simulations and real time implementation of developed controller is shown in Fig. 1.2. The first phase of research is focused on developing the Modified Extended Kalman Filter algorithm for alleviating the external magnetic effects on vehicle yaw estimations. The algorithm is tested for several test path conditions and the results shows that the fusion of vehicle kinematics in IMU measurements improved the vehicle yaw estimation and detailed results are furnished in Chapter. 2. This Chapter mainly

deals with various sensor noise and their effect on vehicle yaw estimations and restricted to individual sensor studies. However, the application of developed modified EKF algorithm for autonomous vehicle path tracking is included in third phase of research work, which is explained in Chapter. 4. Further, the second phase of research laid out base platform for systematic design of controller development for autonomous navigation, which is detailed in Chapter. 3 . Though, the developed Non-linear MPC algorithm in Chapter. 3 is not directly implemented in third phase of work due to computational burden, but the methodology used in the work has been replicated in third phase of work. This includes, the processing of vehicle states, sensor measurements, and controller actuation for autonomous navigation. The implementation of second phase of work on actual vehicle requires high processing systems and it is left for the future work.
Therefore, the third phase of research exploits the developed algorithms from modified EKF and non-linear MPC methodology and implemented on actual $1/5^{th}$ truck using LQG controller.

1.2 Goals and Objectives

As mentioned,the objectives of present work is three fold and it is explained as follows,

1.) Developing sensor fusion techniques that can handle sensor noise and reject considerable environmental disturbance

2.) Developing algorithms that would simulate autonomous vehicle navigation through unstructured environment by avoiding stationary and moving obstacles

3.) real time implementation of LQG controller for path tracking on $1/5^{th}$ truck using GPS base station, IMU and Klaman observer.

1.3 Organization of work

The chapter-1 provides background and motivation for the present thesis work and chapter-2 details about sensor fusion using Extended Kalman filter for estimating the vehicle heading. Further, this chapter-2 provides comprehensive literature review, methodology, experimental setup, test conditions and test results for the sensor fusion techniques. Chapter-3 explains NMPC controller development for off road unstructured environment to avoid obstacles and road grade by maintaining the vehicle safety from rollover. This chapter details about literature, methodology for stationary and moving obstacles, and simulation results for stationary and moving obstacle avoidance by preventing the vehicle from rollover. Chapter-4 includes the real implementation of LQG controller on $1/5^{th}$ truck. This chapter details about experimental set up, that includes interfacing IMU,speed,GPS and Leddar One sensors with Beagle Bone Black embedded system. Further, the chapter provide state estimation studies using kalman filter is explained. Chapters 5 and 6 provide conclusions and future work based on the current research work. Finally, Appendices provide supporting material including Quaternion definitions, MATLAB codes for NMPC controller development and various sensor properties.

Chapter 2

Sensor Fusion studies for vehicle yaw estimation

The vehicle orientation estimations are important in autonomous navigation field [2], vehicle safety [3] and driver assistance technologies. In case of lateral controller development for autonomous navigation, yaw estimations are critical to track the given path [4]. However, the yaw estimates are prone to error with external magnetic disturbances. In the current paper work, we use UM7 3rd generation MEMS based inertial measurement unit for estimating vehicle attitude and heading/yaw. The current sensor consists of 3-axis accelerometer, 3-axis rate gyro and 3-axis magnetometer. Further, the MEMS based inertial unit has advantage of being low cost and compact in size to make easy for installation on vehicle. With appropriate installation and calibration of each sensor on vehicle body can eliminate misalignment errors between sensor co-ordinates and vehicle body coordinates. That said, the current UM7 sensor can provide accurate geomagnetic field, angular rates and gravity vectors in body coordinates in undisturbed, magnetic free environment [5, 6]. The orientation of a body in 3D space can be defined with Euler's angles including, roll, pitch and yaw. Further the combination of roll and pitch is called attitude and yaw is also called the heading. In the present work, we solve for vehicle orientation and deduce the equation for yaw estimations from it. In general, the orientation of a body in 3D space can be estimated by integrating the gyro rate outputs. However, the gyroscope data suffers from drifting phenomena and result would accumulates error with time [7]. This Wahba [8] problem can be solved by fusing the information from accelerometer and magnetometer outputs. However, the accelerometer and magnetometers are prone to motion acceleration and external magnetic disturbances [9, 10]. Therefore, to make accurate orientation estimations, many fusion algorithms have been employed over the years. Out of which two major categories are, complementary filters and

Kalman filters. Complementary filters perform fusion in frequency domain [11], [12], whereas the Kalman filters follows stochastic approach [13, 14, 15]. The current work employs later approach, which has two basic steps including prediction and update step. Prediction step uses gyroscope data to propagate the orientation called priori estimation. In update step, the magnetometer and accelerometer outputs used for correcting the above priori estimations called posterior estimation.
However, ground vehicular models are highly non-linear and conventional Kalman filter estimates are not adequate. The most common approach is to employ Extended Kalman Filter [6], which deduce the non-linearities by linearizing the current state estimates. The steps after linearization follows the conventional Kalman filter, which is explained in the above section. But, there are other approaches including Unscented Kalman filter [14], which handles severe non-linearities and computationally heavier for practical implementation. In this work, we have employed EKF for estimating the vehicle orientation and making special case for handling yaw estimations in magnetic disturbance environments. There are numerous algorithms in the current literature to handle vehicle orientation and each study would make different strategies in terms of defining state vectors, formulating filter structure etc. However, the core concept behind all these methodologies should be same. i.e., accelerometer would be used for correcting the attitude estimations and magnetometer is responsible for correcting yaw estimations. More clearly, the accelerometer only provides attitude information and the filter has to use magnetometer information for correcting the yaw estimates. Therefore, the major problem in estimating the yaw is that, it suffers from various magnetic disturbances including hard iron effects, soft iron effects and environmental magnetic disturbances [5]. The current UM7 inertial measurement unit has the capability to eliminate hard iron and soft iron effects by the magnetometer sensor calibration [5, 16]. However, the external magnetic disturbances can't be compensated due to its high uncertainty in nature [17]. Therefore, in the following context the magnetic disturbance refers to the external or environmental magnetic disturbance alone and it does not include soft and hard iron effects. The main motivation of this paper work is to alleviate external magnetic disturbances on yaw estimations by taking advantage from vehicle kinematics. Therefore, the present paper work results are applicable for ground vehicular applications and not recommended for aerial applications.
Many studies have been made to handle the magnetic disturbances; From studies [17, 18, 19], used magnetic disturbance as a criterion to reduce weightage on magnetometer measurement. However, these techniques would deviate faster as the magnetic disturbances last longer. Further, fusion of different sensors in vehicle orientation estimation not only effect the yaw estimation but also attitude estimations. Thus, the magnetometer sensor outputs would also affect vehicle attitude. To address this issue, some studies restricted the magnetometer output for yaw estimations alone. [9, 10] proposed two-layered structure in Kalman filter by decoupling the quaternion multiplication factors. Suh [20] proposed an indirect Kalman filter with two step

measurement update to restrict the magnetometer outputs only to yaw estimations. In addition, Gang Shi [21] et al. proposed a two-step measurement update method in Kalman filer along with vehicle kinematics for alleviating the magnetic effects on yaw estimations. However, this paper restricts its application to straight line and it is highly insensitive to varying vehicle status, which depends on vehicle kinematics and Coriolis components.
From the above discussion, it can be clearly seen that the magnetic disturbance plays a critical role in yaw estimations. However, this paper takes the advantage of vehicle kinematics, which can provide yaw rate dynamics from completely different perspective. The required parameters for vehicle kinematics calculations are steering wheel angle and vehicle speed, which are clearly independent from magnetic disturbances and prove to be potential solution for the ground vehicle yaw estimations. Therefore, the improved EKF algorithm using vehicle kinematics information is as follows:

1). The time propagation step is unchanged and the priori estimation would be made from gyroscope output.

2). The update step would use normalized geomagnetic field on horizontal plane either from magnetometer or from vehicle kinematics based on the magnetic field disturbance strength. This process would not only alleviate magnetic effects on yaw estimation but also on attitude estimation.

The present work is organized as follows: section 1 (Introduction) explains about various techniques used for estimating vehicle orientation and a comprehensive explanation on Extended Kalman Filter algorithm. Section 2 (Methodology) explains conventional Extended Kalman Filter used in the current work as benchmark filter algorithm and extends to modified EKF algorithm to alleviate external magnetic disturbances for accurate yaw estimations. Further, explanation on vehicle kinematics have been furnished. Section 3 (Experimental set-up) includes vehicle set-up and explanation on noise of each sensor including accelerometer, gyro, magnetometer, steering wheel measurement and speed sensor. Further, calibration of vehicle kinematics calculations with IMU heading measurements has also been provided. Section 4 (Results and Discussion) focus on improved EKF algorithm performance for straight line, 90° turn, round about turn and circle tests. In the end, Section 5 provides conclusions and recommendations for the future work.

2.1 Methodology

2.1.1 Conventional EKF algorithm

The present paper deals with quaternions to represent orientation of the object. It has several advantages over Euler's angle representation including, not suffering from Gimbol lock, low dimensionality and providing a linear formulation of orientation dynamics [9]. That said, in 3D space, any given object orientation with respect to a reference frame can be represented by a unit quaternion q, which is defined as

$$q = \begin{bmatrix} q0 & q1 & q2 & q3 \end{bmatrix}^T$$

Where $q0$ is a scalar part and $\begin{bmatrix} q1 & q2 & q3 \end{bmatrix}^T$ is the vector part of the quaternion. Further, $[b^g_1 \; b^g_2 \; b^g_3]$ is gyroscope bias in X-axis, Y-axis, and Z-axis respectively. The gyroscope handles the quaternion dynamics and it is important to consider gyroscope bias as states of the system to reduce gyro drift errors. Therefore, the Extended Kalman Filter in this section consider the system with seven states including four from quaternion and three from gyro bias.

$$x = \begin{bmatrix} q & b^g \end{bmatrix} = \begin{bmatrix} q0 & q1 & q2 & q3 & b^g_1 & b^g_2 & b^g_3 \end{bmatrix}^T$$

This EKF algorithm works accurately, when there are no external magnetic fields that result in systemic errors. Therefore, this algorithm serves as a benchmark for the modified/Kinematics-fusion EKF explained in the next section. The conventional EKF is an extension of linearized Kalman filter presented in [6]. The derivations of system model equations are given in Appendix.A. The Euler angles including yaw, roll and pitch are computed from these final quaternion estimates.
The present paper considers North, East, Down (NED) as reference frame and Forward, Right and Down as body frame (B-frame) [16]. The Euler angles, yaw, roll and pitch can be chosen from rotations around B-frame Z, Y and X axis respectively. The following section, provides notations used in the rest of the work.
x denotes the system state vector; b^g is the gyroscope bias; m_r and g_r are geomagnetic and gravity vectors resolved in the reference frame; m, w and a represents the 3-axis IMU sensor data including magnetometer, gyroscope and accelerometer respectively; The matrices, C^b_r denotes the rotation matrix from reference frame to body frame; O and I represents the null and identity matrices with their subscripts indicating their dimensions.
Subscripts: x, y and z for a given vector represents the vector measurement in their

respective axes; k represents the current time step; r, denotes reference frame; b, denotes body frame; m, w and a denotes magnetometer, gyro and accelerometer sensor.

2.1.2 System propagation Model

As mentioned, the system state vector consists of quaternion and gyro bias which is defined as,

$$x = \begin{bmatrix} q^T & b_g^T \end{bmatrix}^T \tag{2.1}$$

The propagation model estimates the system dynamics using gyroscope angular measurements and can be written in terms of Quaternion rotation as,

$$\dot{q} = \frac{1}{2} q \bigotimes w \tag{2.2}$$

The Equation. (2.2), can be expanded in matrix form as follows (see Quaternion multiplication in Appendix.A.1 for more details),

$$\dot{q} = \frac{1}{2} S(w) q = \frac{1}{2} S(q) w \tag{2.3}$$

Where, $q \bigotimes w$ denotes Quaternion multiplication, which implies rotation of Quaternion by the amount of w rate in X, Y and Z directions. More details on system dynamics, $S(w)$ and $S(q)$ are explained in the Appendix.A.1. As mentioned earlier, the gyro bias is a part of state vector and needs to be compensated by system dynamics. Therefore, the final system propagation model can then be expressed as,

$$\dot{q} = \frac{1}{2} S(w - b^g) q = \frac{1}{2} S(q - b^g) w \tag{2.4}$$

The above system dynamics model Equation.2.4, is a non-linear continuous model and this needs to be linearized and discretized to apply EKF to the system. A simple first order linearized model can be made using Euler's forward method and is sufficient for this application. A simple first order linear discrete model can be written as follows,

$$\dot{q_k} = (q_{k+1} - q_k)/T \tag{2.5}$$

Where, T, is the iteration time between sample k+1 and k From Equations.2.4 and 2.5

$$q_(k+1) = \frac{T}{2}S(q_k)w - \frac{T}{2}S(q_k)b^g + q_k$$
$$b^g_{k+1} = b^g_k$$

Therefore, the system propagation model along with gyro bias as state vector becomes,

$$x_{k+1} = Ax_k + Bu_k$$

$$\begin{bmatrix} q \\ b^g \end{bmatrix}_{k+1} = \begin{bmatrix} I_{4\times4} & -\frac{T}{2} \times S(q) \\ 0_{3\times4} & I_{3\times3} \end{bmatrix} x_k + \begin{bmatrix} \frac{T}{2} \times S(q) \\ 0_{3\times3} \end{bmatrix} w_k \tag{2.6}$$

Where,

$$A = \begin{bmatrix} I_{4\times4} & -\frac{T}{2} \times S(q) \\ 0_{3\times4} & I_{3\times3} \end{bmatrix}, B = \begin{bmatrix} \frac{T}{2} \times S(q) \\ 0_{3\times3} \end{bmatrix}$$

The complete derivation and matrix manipulations can be seen in Appendix.A.1.

2.1.3 System Measurement model

The measurement model uses accelerometer and magnetometer measurements, with the fact that the gravity vector and geomagnetic north vector are known at a given position in the NED frame. Therefore, the measurement model can be explained into two main sub-models namely Accelerometer model and Magnetometer model

2.1.3.1 Accelerometer model

As mentioned, the accelerometer model assumes the gravity vector is known in the NED frame at a given position. This known gravity vector can be rotated into B-frame using rotation matrix C^b_r to get the acceleration in B-frame. In essence, this calculated acceleration vector is propagated from the previous system dynamics model, Equation 2.6. Finally, in EKF update step, this calculated acceleration in B-frame can be compared with the measured acceleration from accelerometer, which is measured in B-frame. An adaptive weightage has been applied based on the magnitude in the error, which is discussed in the next section. Therefore, the accelerometer model for calculating acceleration in B-frame is as follows,

$$\overline{y}_a^b = C_r^b(a_e - g) + e_a^b + b_a^b \tag{2.7}$$

Where, $\overline{y}_a^b$, is calculated acceleration in B-frame
C_r^b, is rotation matrix from reference frame to B-frame, defined in Appendix.A.1.
a_e, is external acceleration in B-frame
g, reference gravity vector
e_a^b, b_a^b, are accelerometer noise and bias respectively in B-frame
In the present work, the acceleration due to external forces are negligible as the current vehicle speed is less than 3 m/s and the tests are conducted at nearly constant speed. Further, the bias in accelerometer is quantified and compensated through accelerometer calibration, detailed in experimental set-up section. The noise in the accelerometer is modeled as Gaussian white noise and the details of sensors noise is provided in Table.A.1, in Appendix.A.2. Therefore, the adjusted final accelerometer measurement model becomes [6][22],

$$\overline{y}_a^b = C_r^b(-g) + e_a^b \tag{2.8}$$

2.1.3.2 Magnetometer model

The magnetometer can be modelled in a similar fashion of accelerometer. In the present work, magnetometer model works based on the fact that the geomagnetic north vector is known at a given location in reference frame. Madwick [12] uses modified magnetic reference vector to have same inclination as the measurements. However, this approach simplifies the magnetic declination to 0 deg and thus making measurements to correct offset only in measured declination angle. Therefore, to account for magnetic declination and have same inclination as measurements, use known geomagnetic north vector (m_r) for the given location and set its z-axis reference to 0 and is then rotated back to B-frame [6]. This can be formulated in magnetometer model as follows,

$$\overline{y}_m^b = C_r^b(m_r) + e_m^b \tag{2.9}$$

$\overline{y}_m^b$ is calculated magnetic field in B-frame
m_r, reference/known magnetic field vector
e_m^b, magnetometer noise in the B-frame
Similar to accelerometer, the magnetometer noise is modeled using Gaussian white

noise. The calibration of each sensor and their scaling and bias values are given in experimental section.
From equations 2.8 and 2.9, both accelerometer and magnetometer models are non-linear, and it is required to linearize the models to use it in the EKF implementation. The details of linearization have been given in Appendix .A.2. The final measurement model to implement in EKF filter becomes,

$$\overline{y} = \begin{bmatrix} \overline{y}_a^b \\ \overline{y}_m^b \end{bmatrix} = Cx_k \tag{2.10}$$

Where,

$$C = \begin{bmatrix} C_a & 0_{3\times3} \\ C_m & 0_{3\times3} \end{bmatrix}, x_k = \begin{bmatrix} q \\ b^g \end{bmatrix}_k$$

Here, C_a and C_m are Jacobian matrices, results from linearizing the accelerometer and magnetometer models and is detailed in Appendix.A.2. The bias term in magnetometer is neglected due to the assumption that, the magnetometer is well calibrated before the tests. However, a brief explanation on magnetometer calibration is provided in results section.

2.2 Adaptive EKF Filter

By using the propagation and measurement model equations 2.6-2.10, the adaptive EKF filter for the given IMU sensor measurements can be implemented as follows:

2.2.1 Prediction step

Provide initial values of filter state estimates including $\hat{x_k}$ and $\hat{P_k}$ [21] based on sensor data and in the present work, process noise Q has been considered as simple piecewise noise to reduce the computational burden on implementation.

$$\overline{x}_{k+1} = A\hat{x_k} + Bu_k \tag{2.11}$$

$$\overline{P}_{k+1} = A\hat{P_k}A^T + Q_k \tag{2.12}$$

The matrices A,B and C are already defined in the previous sections. Therefore, the propagation of prediction step from equations 2.11,2.12 are called priori estimations at k+1 step, can be computed using gyroscope output, and posterior estimations at step k. Here, the posterior estimations are computed from update step, after the initial iteration is completed through initial guess estimates. c

2.2.2 Update step

The outcomes of update step is called posterior estimations and these can be calculated using the outputs from magnetometer and accelerometer sensors, measurement models and priori estimations at k+1 step:

$$K_{k+1} = \frac{\overline{P}_{k+1}C_{k+1}^T}{C_{k+1}\overline{P}_{k+1}C_{k+1} + R} \tag{2.13}$$

$$\hat{x}_{k+1} = \overline{x}_{k+1} + K_{k+1}(y_{k+1} - C_{k+1}\overline{x}_{k+1}) \tag{2.14}$$

$$\hat{P}_{k+1} = (I_{7\times7} - K_{k+1}C_{k+1})\overline{P}_{k+1} \tag{2.15}$$

Here, R is a measurement noise vector to provide weightage for measurements and K is a Kalman gain vector to distribute weightage between model propagation and measurements. As mentioned earlier, the state vector x_k consists of 7 states and thus to maintain dimensionality, it is required to append 3×3 null matrix to the measurement model is shown in equation. 2.10. After the update step, the quaternion in posterior estimations need to preserve its unit-norm property. Therefore, the updated quaternion in state vector needs to be normalized for recursive EKF implementation.

$$q_{k+1} = \frac{\hat{q}_{k+1}}{\|\hat{q}_{k+1}\|} \tag{2.16}$$

The Equation. 2.16, provides normalized quaternion, which can be used for calculating the Euler angles including yaw, roll and pitch using rotation matrix C_r^b [21][19]. Finally, the posterior estimates from equations 2.13-2.16 serve as inputs for the priori estimations given in equations 2.11-2.12 and evolves the filter in time zone recursively.

2.2.3 Modified/Kinematics-fusion EKF

As mentioned above, the conventional EKF filter works accurately, when the system is free from external magnetic disturbances. In order to alleviate the effect of external magnetic disturbance, the conventional EKF filter is integrated with the vehicle kinematics information. This fusion of vehicle kinematics information with conventional

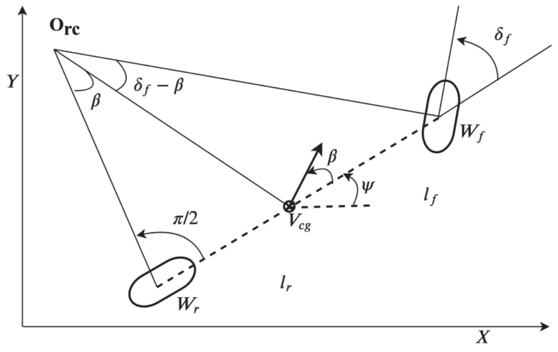


Figure 2.1: Schematic of vehicle lateral kinematics motion with front wheel drive system.

EKF provide robust vehicle orientation estimations and avoid sudden jerks in the autonomous navigation. The above conventional EKF has an advantage of keeping magnetometer model separated from accelerometer and for some extent, the effect of external magnetic disturbance only changes the yaw value in orientation estimation [6] by making reference magnetometer vector Z-axis value to 0. Further, the present EKF approach normalizes the magnetometer measurements in the update step. This normalized magnetometer measurement vector can be replaced with the vehicle kinematics yaw information by applying a simple trigonometric rule, during the external magnetic field disturbance.
The modified EKF algorithm keeps prediction step unchanged. The update step requires knowledge from vehicle kinematic model. This is briefly explained in the following section. The lateral kinematics [22] model assumes bicycle model with slip angles at both wheels are zero. From [22], this is a valid assumption for vehicle speeds up to 5 m/s and further, the vehicle consists of front wheel drive Ackerman steering mechanism with ratio from steering wheel to road wheel given in Equation. 2.20. Therefore, the rear steering wheel angle δ_r for the current vehicle is considered zero. From the trigonometric rules and Fig.2.1, the equation for yaw rate becomes [22],

$$\dot{\Psi} = \frac{V\cos(\beta)}{l_f + l_r}\tan(\delta_f) \tag{2.17}$$

Where, vehicle slip angle

$$\beta = \arctan(\frac{l_r\tan(\delta_f)}{l_f + l_r}) \tag{2.18}$$

Here O_{rc} is instantaneous rolling center and V_{cg} is vehicle center of gravity. In addition, for a typical Ackerman steering geometry the average front steering angle

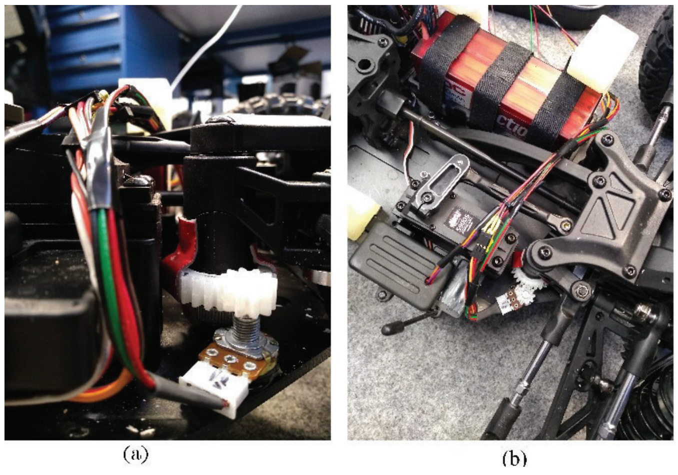


Figure 2.2: 1/5th vehicle front steering angle measurement set up.

can be calculated as [22][20],

$$\delta_f = (\delta_o + \delta_i)/2 \tag{2.19}$$

Where, δ_o is outer front wheel steering angle and δ_i is inner front wheel steering angle. From Equation.2.17, it is clear that the yaw information from vehicle kinematics can be obtained by measuring the vehicle speed and steering angle, and it is explained with appropriate experimental setup in the following section. Fig.2.2 shows the steering angle measurement using potentiometer, which is connected to the steering link of the vehicle though a 3D printed gear system. The on-board embedded system reads potentiometer position and makes ADC conversions to calibrate the steering system. Here, steering system is calibrated with 0 Volts being the left most steering angle and 3.3-Volts being the right most steering angle position. However, the current vehicle has total RWA range of $\pm$ 27.25° with -ve angle being left turn and vice versa. From the calibration analysis, it is found that the ratio between steering wheel angle to road wheel angle as,

$$\delta_f = 0.0253 \times RWA + 0.5 \tag{2.20}$$

In addition, the noise in RWA is estimated as $\pm$ 0.5 deg for the total range of $\pm$ 27.25°. On the other hand, vehicle velocity is measured using a Metallic-Object Proximity Switch sensor and from the sensor measurement analysis it is found that the sensor can able to measure speed from 0.5-5 m/sec with an accuracy of $\pm$0.01 m/sec. Due to its low variance/noise in vehicle speed and RWA measurements, the vehicle kinematics restricts yaw rate estimations to within $\pm$ 1.25 deg/sec variance. Further validation of vehicle kinematics yaw calculations is given in experimental setup section. By knowing the yaw dynamics at current step k, from vehicle kinematics and iteration

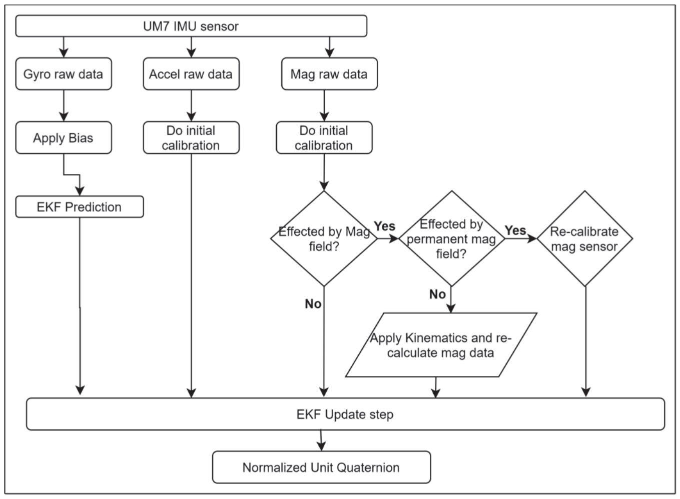


Figure 2.3: Modified vehicle kinematics EKF fusion algorithm.

time dt provides yaw estimation at k+1 step as,

$$\Psi_{k+1} = \Psi_k + \dot{\Psi}_k(dt) \tag{2.21}$$

The key point to note here is that, the vehicle kinematics provide yaw estimations based on Cartesian coordinate system, whereas the EKF filter works based on global coordinate system. Due to this, in the present work, yaw measured in global co-ordinates by IMU is converted into Cartesian coordinates (CCW) and thus maintain consistency between vehicle kinematics calculations and IMU heading values for the fusion algorithm. Therefore, all heading results interpretation is made based on Cartesian co-ordinates, where East direction starts from zero degrees and rotates in CCW direction for 360 degrees. Yaw estimation from the above simple first order equation. 2.21 can be mapped into magnetic x and y fields by applying a simple trigonometric rule,

$$m_x = \cos(\Psi_{k+1}) \tag{2.22}$$
$$m_y = \sin(\Psi_{k+1}) \tag{2.23}$$

The equations 2.22-2.23 assume that the magnetometer is calibrated and its norm is a unit circle on horizontal plane. The updated magnetometer values from equations 2.22-2.23 would replace the erroneous magnetometer readings, while the system is subjected to external magnetic disturbance is shown in Fig.2.3. The above model shows robust performance not only during straight path but also in turning paths.

2.3 Experimental setup

Experiments were conducted at APSRC, Michigan Technological University and the vehicle used for the tests is 1/5th scale buggy type truck. The vehicle has been equipped with the UM7 inertial measurement unit, potentiometer along with 3-D printed gear to measure the steering angle, speed senor and on-board embedded system for acquiring data. A fixed ratio of 12.6 has been applied between center axle shaft to wheel speeds. The data logged at a rate of 80 Hz from each sensor and 1.2 GHz ARM cortex Embedded processor has been used for logging and processing the data.
The filter initial estimates were taken from UM7 gyro, accelerometer and magnetometer sensor outputs. All three sensors are calibrated before doing the tests and assumed that soft and hard iron effects on magnetometer sensor is eliminated [5, 16]. The calibration and validation procedure for accelerometer and vehicle kinematics is as follows:

2.3.1 Accelerometer calibration

Assuming that, the sensor axes are aligned with the Body axes, the relationship between accelerometer measurements from sensor frame to body frame can be written as,

$$a_s = K_a a_b + b_a \tag{2.24}$$

Where, a_s is acceleration measurements in sensor frame. The bias and scale factor in each axis made the above model to have 6 unknowns. These 6 unknowns can be calculated by placing IMU in vertical position in their corresponding axes. i.e, for calculating bias and scale factor in x axis, the IMU x-axis would be placed in vertical direction. This way, the accelerometer x component measures known gravity component and provides two equations for x-component unknowns. Similarly, by including y and z axis components it formulates six equations and with 6 unknowns and therefore the scaling factor and bias matrix for the given UM7 accelerometer is estimated as,

$$b_a = \begin{bmatrix} 58.85 & -38.25 & 0.012 \end{bmatrix}^T$$

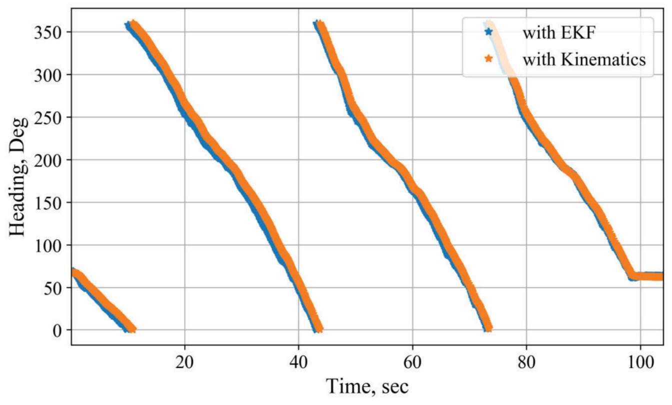


Figure 2.4: Heading comparison between IMU sensor measurements and vehicle kinematics calculations.

Scaling matrix or coefficient,

$$K_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4107.0 \end{bmatrix}$$

In the present work, Gyro sensor bias terms are included in the state vectors of Extended Kalman Filter. Therefore, the filter would compensate gyro bias for each iteration and the calibration of gyro for bias is not required. The calibration of Magnetometer is briefly discussed in the results section.

2.3.2 Vehicle kinematics validation

It is important to validate the vehicle kinematics calculations according to the IMU sensor measurements such that the fusion algorithm provide accurate heading estimates. To confirm the kinematics calculations for accuracy, a predefined circular path test condition has been made. The circular path is made with 5-meter radius and vehicle is travelled three times along this path without any magnetic effects and thus conventional EKF estimates from IMU can be considered reference values for the heading comparison.
Fig. 2.4 shows the comparison between heading estimates from IMU EKF filter and heading calculations from vehicle kinematics for three circles in a test. The error between IMU heading and vehicle kinematics heading at the end of three-circle test is within $\pm 1.5°$ and therefore the heading estimation from vehicle kinematics can be fused with IMU when it is affected by external magnetic disturbances. Further the true distance measured for the circles is $94.2m$ and the distance calculations from the

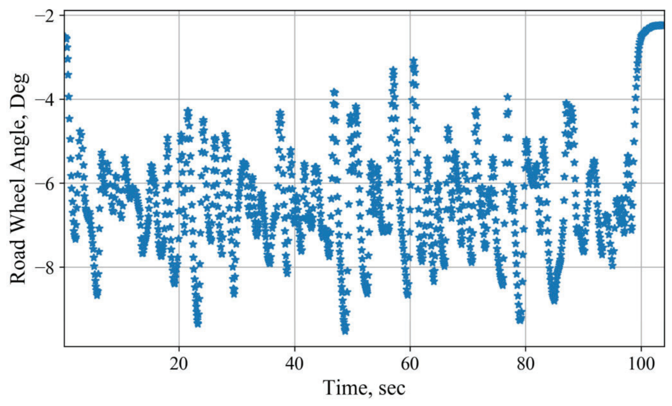


Figure 2.5: Road wheel angle measurements for circular path test.

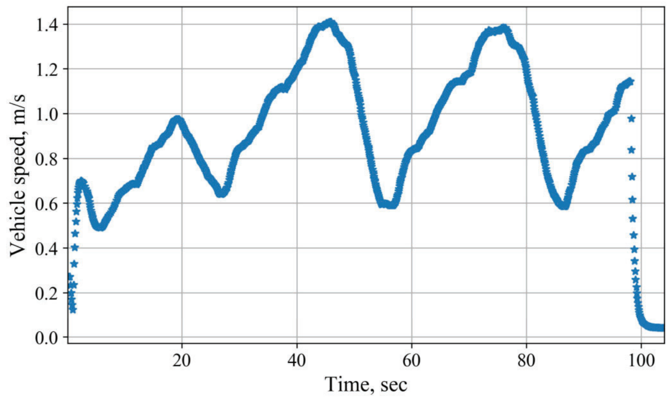


Figure 2.6: Vehicle speed measurements during circular path test.

integration of measured vehicle speed obtained is 93.8 m. In addition, a simple low pass filter has been applied to the velocity and steering measurements for rejecting outliers in the data. The processed Road wheel angle and vehicle speed data used for the vehicle kinematics calculations in circle test are provided in Figures 2.5 and 2.6.

2.4 Results and Discussion

As mentioned, all the tests were made after calibrating three sensors in IMU including accelerometer, gyro and magnetometer. The Norm of magnetic field is calculated as follows

$$Norm\ of\ magnetic\ field, \zeta = \sqrt{m_x^2 + m_y^2}$$

The IMU orientation values are prone to magnetic errors as the magnetometer is sensitive to ferrous and permanent magnets. In the present paperwork, the magnetic field is mainly divided into two main categories namely temporary and permanent deformation states based on the Norm of magnetic field. If the Norm of magnetic field exceeds certain limit for certain time period, the magnetometer sensor experiences permeant deformation in its readings and needs recalibration to estimate the accurate vehicle orientation. However, studying different types of magnets and controlling its magnetic field direction is out of this scope of work. In the present work, the Norm mag field ζ, provide limitations for temporary deformation and adaptive weights for vehicle kinematics data in EKF fusion algorithm. Further, yaw rate and rate of change of Mag norm is also considered as a supplemental information for vehicle kinematics fusion algorithm. The criteria to fuse vehicle kinematics information with conventional EKF filter is explained in straight-line static test section.
The current study has conducted mainly two types of tests namely static and on-road tests. The static tests are for simulating the straight road conditions while keeping the vehicle stationary and all the other functionalities are in working condition. This can simply be achieved by keeping the vehicle wheels above ground level and observing the IMU, steering and speed sensor values. These tests are useful for estimating the calibration parameters for gyro, accelerometer and magnetometer sensors and studying the effect of temporary and permanent magnetic field on magnetometer. Further these tests provide initial guess values for EKF filter algorithm including q_0 and b_0 using magnetometer, accelerometer and gyro outputs. The initial values for parameter P_0 have been considered as $100 \times I_{7\times7}$ and Q, R_a and R_m matrices are calculated from q_0 and P_0 [6].

2.4.1 Static tests

As mentioned, these tests are conducted by lifting the wheels off from the ground and keeping the vehicle heading towards known direction with true value of yaw is 163° in Cartesian coordinates. The test has been made for about 175 sec period and the temporary magnetic field is applied during 50-110 sec time duration. During this period, the Norm magnetic field is deviated about 0.8 Mag fields from unit norm circle is shown in Fig.2.8. This deviation in Norm magnetic field indicates the external magnetic field or disturbance on IMU and thus, it clearly affects the vehicle yaw estimations is shown in Fig.2.7. However, the sensor reinitializes to its original state and measures correct yaw values, once the external magnetic field is removed.
From Fig.2.7 it can be seen that, when the system undergoes external magnetic disturbance (between 50-115 sec duration), the kinematics-fusion algorithm keeps the yaw estimations unchanged as the change in steering wheel angle is about zero degrees. However, this fusion of vehicle kinematics is made with the interpretation

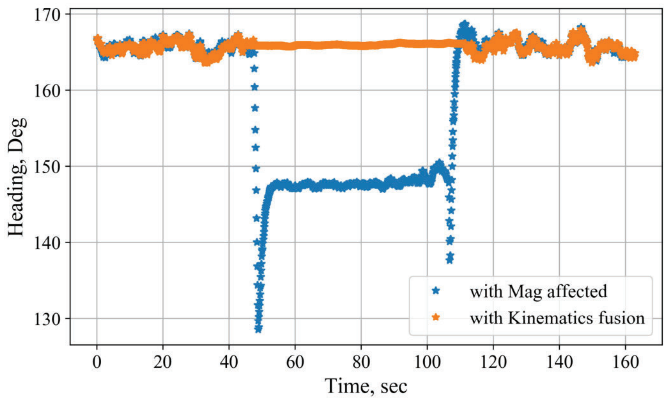


Figure 2.7: Comparison of heading estimates with and without kinematics fusion for static test condition.

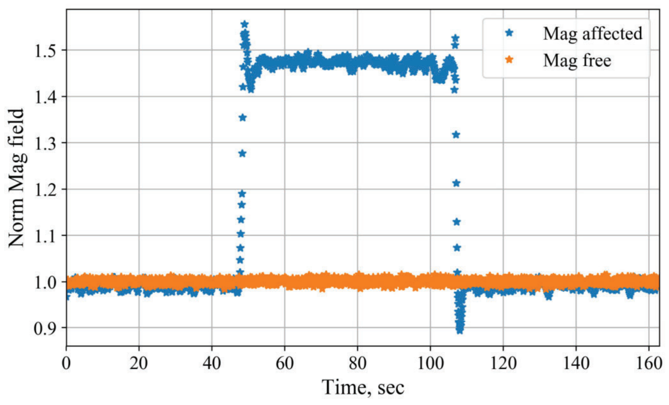


Figure 2.8: Magnetic strength comparison for static test condition.

of different sensors data, which is explained in criteria to apply kinematics fusion algorithm section.

2.4.1.1 Criteria to apply Kinematic fusion algorithm

The application of kinematics fusion algorithm for heading estimates not only depends on Norm of mag field but also on other parameters including rate of change of norm mag field, accumulation of norm mag field and rate of change of heading or yaw rate. From Fig.2.8, the primary criteria to apply kinematic fusion algorithm is to check Norm mag field is within the range of ± 0.05 from unit Norm circle. Therefore, any Norm Mag field data outside this range initiates the fusion of vehicle kinematics in the algorithm. From Fig.2.9, it can be seen that, the rate of change of Norm of

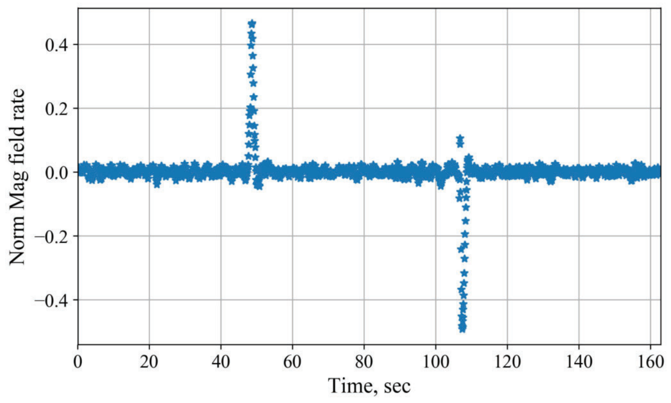


Figure 2.9: Rate of change of Norm of mag field for static test condition.

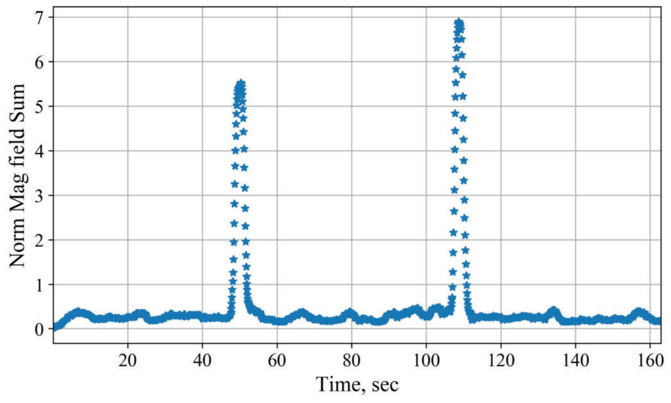


Figure 2.10: Norm mag field sum for static test condition.

mag field only changes when there is abnormal change in the magnetic field. This parameter clearly indicates the effect of external magnetic disturbance and thus avoid sensor estimates in the adaptive kinematics fusion algorithm. This parameter sets the criteria to accept sensor estimates only when the rate of change of Norm mag fields are within ± 0.03.

From Fig.2.10, The Norm mag field Sum is calculated by integrating the previous 40 samples of Norm Mag field rate values and thus making sure that, the sensor re-initialization has been made after the initial disturbance from external field. This effect can be seen in Fig.2.23, where the IMU is still under mag-affected zone while Norm mag field passes through acceptable range. This condition indicates robustness to different directions of external magnetic field on IMU and thus providing stable estimates for vehicle orientation. This parameter makes criteria to accept sensor estimates only when the sum is below 0.5. The final criteria to check the disturbance of sensor estimates for heading is to have stable and smooth yaw rate values. From the vehicle steering mechanism perspective, it is known that the vehicle has constraints for steering rate and thus impose a check point for abrupt changes from sensor yaw rate

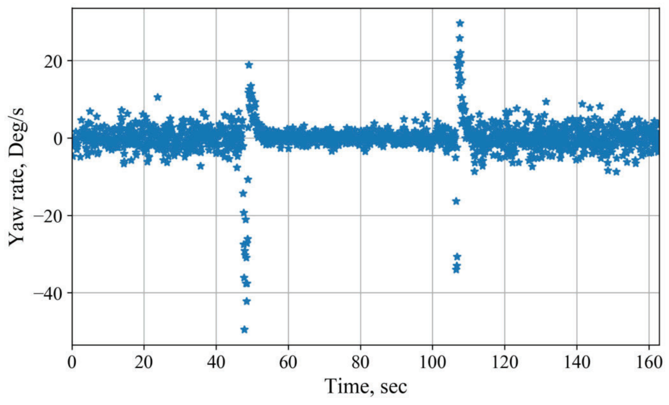


Figure 2.11: Yaw rate variation for static test condition.

estimates. This parameter makes criteria to accept sensor estimates when the absolute yaw rate is within $20°/sec$ is shown in Fig.2.11. From the above criteria analysis, the fusion algorithm rejects sensor measurements whenever there is a possibility of external magnetic disturbance and waits for the sensor to stabilize its measurements after passing external disturbance region. Therefore, this provides smooth transition between Mag-affected regions to Mag-free region with no abrupt changes in steering command for autonomous navigation.
When the applied external magnetic field deviates about 3 units from unit Norm circle for about 50 secs, the magnetometer undergoes permanent deformation and it requires recalibration for the given location [5][21]. The next section provides brief explanation on magnetometer recalibration for the given location.

2.4.1.2 Recalibration studies

Fig.2.12, shows the amount of magnetic deviation in magnetometer measurements from unit norm circle. This clearly shows the need for magnetometer recalibration and therefore magnetometer has been re-calibrated using Kok [5] algorithm. To explain briefly on calibration procedure, the IMU has been rotated arbitrarily to obtain 3-dimensional mag vectors for at least 700 sec [16]. This procedure ensures acquiring sufficient mag values from X,Y and Z coordinates and calibration algorithm take these values for finding magnetic errors in each axis through $A^(-1)$ and b_m matrices to transform deviated Norm field onto unit Norm sphere is shown in Fig.2.13.
Here, $A^{(-1)}$ accounts for misalignment, scaling and soft-iron errors, whereas b_m accounts for bias and hard-iron errors [5]. The final magnetometer calibration model

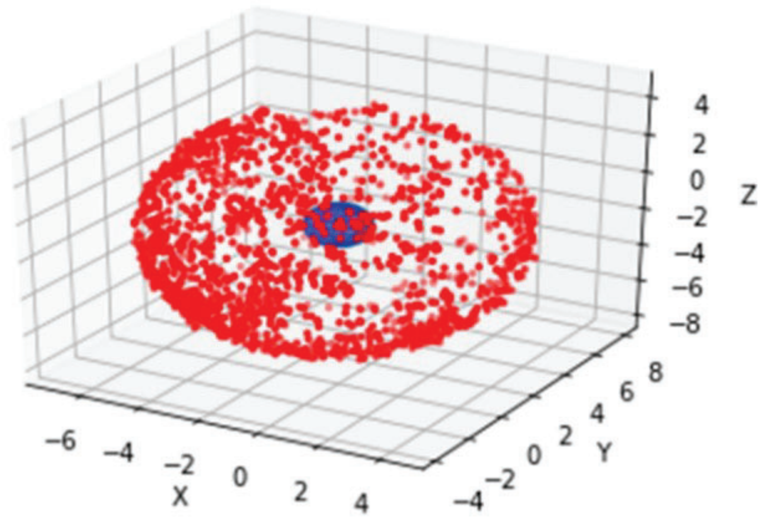


Figure 2.12: Norm Magnetic field before the calibration.

can be written as follows,

$$B_{act} = A^{-1}(B_{meas} - b_m)$$

Where, B_act is actual magnetic field and B_meas is erroneous measurements from magnetometer. The matrices $A^{(-1)}$ and b_m for the present UM7 magnetometer is calculated as,

$$b_m = \begin{bmatrix} 27 & 4 & -35 \end{bmatrix}^T$$

$$A^{-1} = \begin{bmatrix} 0.01386984 & -0.0002846658 & -4.56969E-05 \\ 0 & -0.01405845 & -8.17955E-06 \\ 0 & 0 & 0.01322711 \end{bmatrix}$$

This way calibration of magnetometer has been done and Fig.2.13 shows the Norm Mag field after recalibration of magnetometer for the given location.

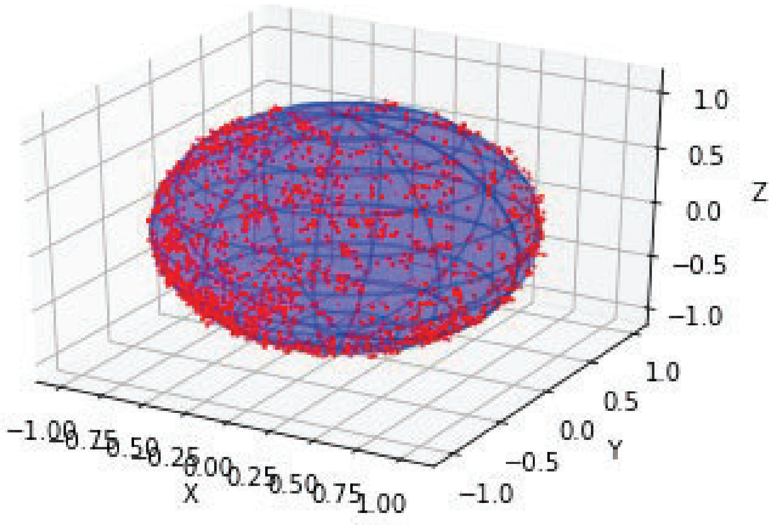


Figure 2.13: Norm Magnetic field after the calibration.

2.4.2 On-Road Tests

2.4.2.1 Test conditions

The vehicle tests have been conducted for various turning conditions including straight road, 90 deg turn and 180 deg turns. The 180 deg turn mimic the round-about road condition and 90 deg turn mimic the left turn respectively is shown in Fig.2.14. Appropriate road turning radius have been made to navigate the vehicle in smooth manner along turnings and reduce error in test to test variation data. In the present study, each test condition is done twice to obtain Mag-free data from one test and Mag-affected data from another test. Further each Mag-free and Mag-affected test condition is repeated four times. The details on total test condition matrix and corresponding Mag-affected zones is given in Table.A.2, Appendix.A.5. All tests are conducted in open environment and magnetic disturbance from ferrous material or any other source has been avoided. Various static tests have been conducted to study the effect of specific materials that can disturb IMU magnetometer in the constrained magnetic field window of 1 ± 0.8 Norm of mag field. These specific materials are used for disturbing the IMU and applied in a strategic manner during the tests. The dots in Fig.2.14 indicates corresponding Mag-test condition check points for applying external magnetic field disturbance on IMU. For instance, while doing 90 deg east to north Mag-tests external mag field is applied at point N2 (7 m from starting point) and released at point N1 (8 m from end point) with total mag-affected zone distance

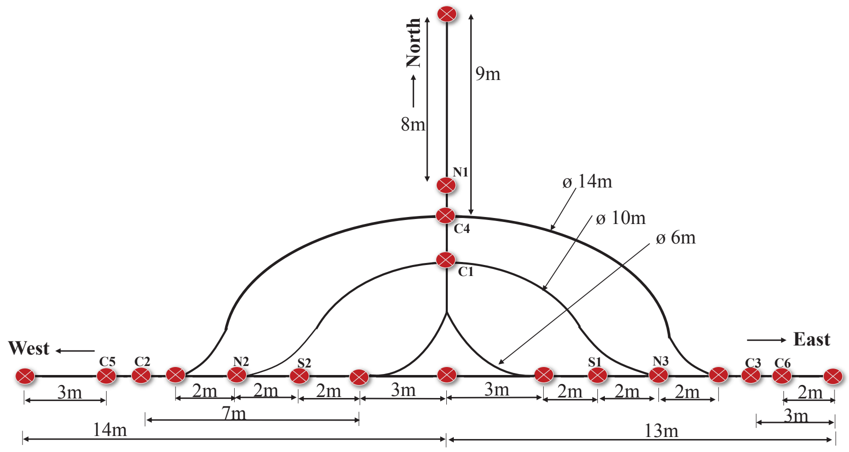


Figure 2.14: Vehicle path for different on-road test conditions.

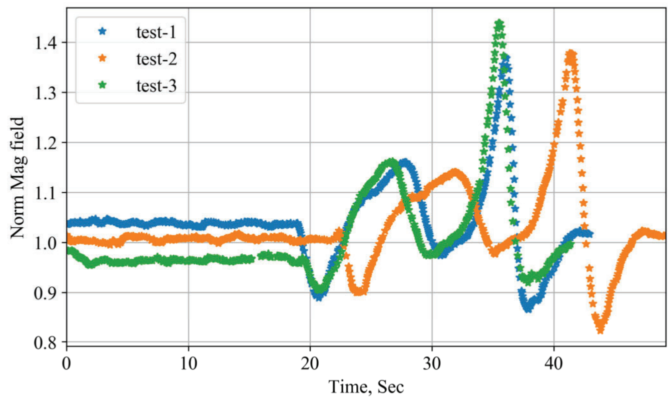


Figure 2.15: Change in Norm of Mag field, when vehicle passes through magnetic prone region.

of about 14m. This way, the effect of external magnetic field for each turning path is made and developed kinematic fusion algorithm. More details on typical range of external magnetic disturbance from environment is explained in next section. In all of the test conditions, vehicle speed is controlled by the controller and steering has been controlled manually.

In general, it is important to know the magnitude of external magnetic field or disturbance in order to apply appropriate filter conditions in vehicle kinematics fusion algorithm. For this reason, we have run the vehicle in a straight line near magnetic prone region, where the amount of external magnetic field is high and would mimic typical outside on-road magnetic disturbances.

Fig.2.15 and 2.16 shows the typical external magnetic field range and change in heading due to this external disturbance. From Fig.2.16, it can be observed that, when vehicle is in mag-free zone (0-20 sec period) its heading is nearly constant and Norm

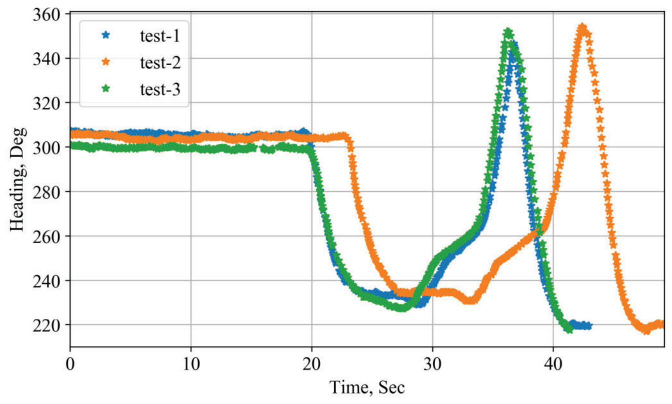


Figure 2.16: Change in heading, when vehicle passes through magnetic prone region.

mag filed is within the range of ±0.05 from Unit Norm value, and as the vehicle approaches near magnetic prone region at about 20th second, the heading values deviates from its true value and Mag norm fields varies suddenly and oscillates about Unit Norm field. Though the results show Norm Mag field range is between 1± 0.4, the current study increased this limit to 1±0.8 to encapsulate stronger and noisy magnetic fields and thus smooth variation in kinematics-fusion algorithm estimates. Therefore, in the present work all test conditions were applied within 1± 0.8 unit norm external mag field to mimic real world road conditions. Further, previous studies [21][19] constrained the fusion of vehicle kinematics information to straight line navigation. However, the current study explored the fusion of vehicle kinematics for various turning paths including 90 deg turns, round-about path and circle path.

2.4.3 Studies on Straight line tests

A straight-line path has been made perpendicular to the benchmark building wall. The true or reference heading has been confirmed with digital compass and the error in true heading due to road slope and vehicle navigation is ± 2.5 deg. However, the true or reference heading for a given test condition is calculated by averaging the 4 mag free test conditions data sets is shown in Fig.2.17.
As mentioned, the three sensors in IMU are calibrated before doing the tests. The tests are conducted in a fashion that, first 4 tests are conducted without any magnetic disturbances on IMU and remaining 4 tests are conducted with external magnetic field disturbance for a fixed duration. Therefore, in mag-affected tests external magnetic field is applied only for a specified distance to observe the effect of magnetic disturbance and recovery of IMU after removing the external magnetic field in the same

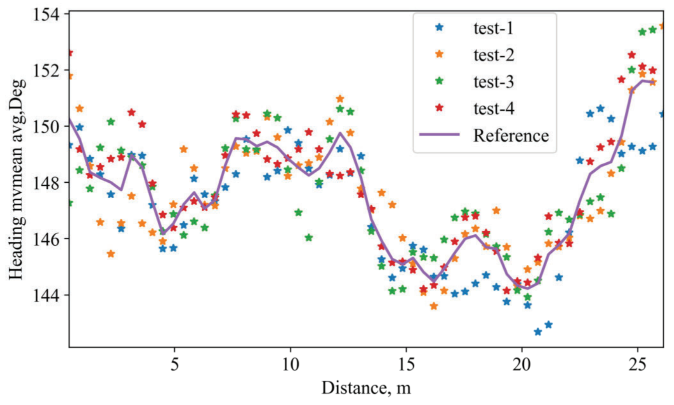


Figure 2.17: Reference heading for the Straight-line navigation test.

test.
There are mainly two tests conducted in straight line tests namely towards-West and towards-East direction. For each direction, the results show that, fusion of vehicle kinematics improves the accuracy in heading estimations and alleviates the effect of external magnetic field is given in Table 2.1. Most of analysis plots are made with respect to the distance travelled by the vehicle. However, data acquisition for each test occurs at different measurement rate and comparison between test cases made using moving mean average in the given interval of time. i.e., the data in every 0.45 sec interval of time period has been averaged to make one point and compare it to the next test case. This way every test case maintains data with 0.45 sec interval period. Fig.2.17 provide reference or true heading for the present straight line to west test condition by averaging the four mag-free test data sets. This reference curve acts as true path for the mag-affected tests and make further analysis on error estimations. In addition, the test to test variation error for each test is estimated by calculating average of RMS errors of four mag-free heading data sets with respect to reference heading is given in Table.2.1. For straight line towards-West test, the test to test variation is estimated as 0.08° From Fig.2.18 and 2.19, it can be observed that, for the given location, Norm Mag field range of ± 0.05 provides no error in the heading measurement and as the magnetic norm field deviates from its calibration circle, the heading value departs from the true value. Further, the decrease in magnetic field from its unit circle has more effect on heading disturbance compared to increase in magnetic field. Fig.2.20, shows magnitude of least square heading error deviation along the path and as expected, the deviation is maximum during the mag disturbance zone. The kinematic fusion estimates show slight deviation from the true data and thus falls near zero line. The improvements in kinematics fusion estimates compared to the conventional algorithm estimates can be quantified using RMS error, which reduces from $3.4 to 0.5°$ without considering the test to test variation error of 0.08° is shown in Table. 2.1. This confirms kinematics fusion algorithm alleviate the effect of magnetic disturbance and thus provide stable heading estimates to avoid

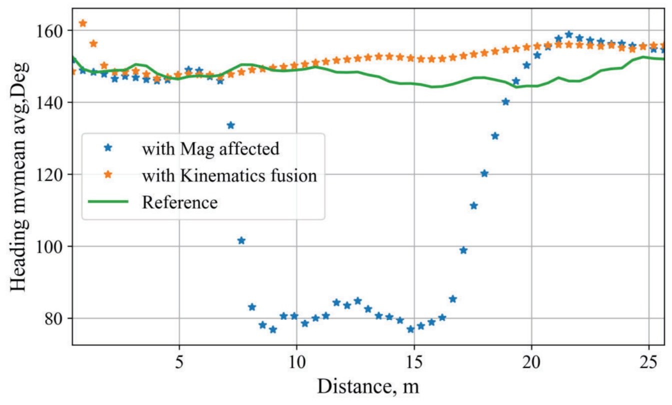


Figure 2.18: Comparison of vehicle heading with and without kinematics fusion algorithm for straight line test condition.

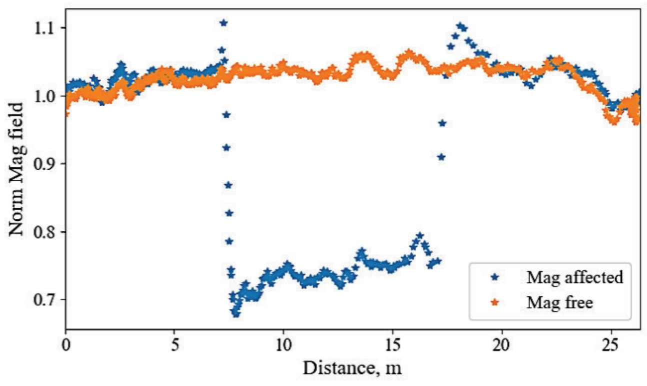


Figure 2.19: Comparison in Norm of Mag field for straight line towards West test condition.

sudden changes in the autonomous navigation.

2.4.4 90 Deg turn road path analysis

The main focus of this paper is to estimate vehicle heading during turning paths while it is affected by the temporary magnetic field disturbance. This can be achieved by taking the advantage of vehicle kinematics and thus it is only applicable to ground vehicular applications. Further, vehicle speed during tests is below 5 m/s and thus vehicle lateral dynamics can be ignored [5].
In this test, the vehicle travels from East to North direction by taking 90 deg left turn with the radius of 3m and the total distance travelled by vehicle is about 27

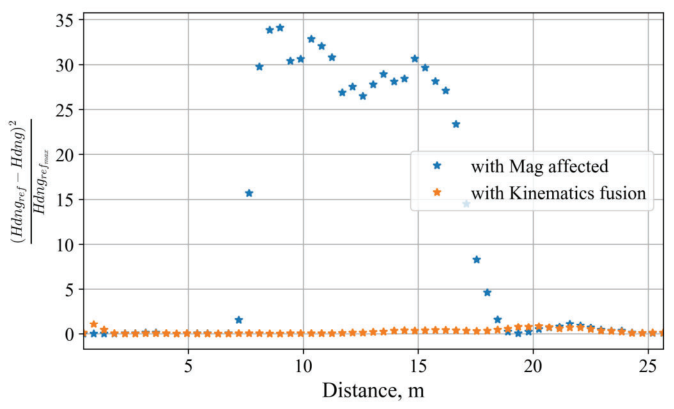


Figure 2.20: Least square error change along straight line towards West path.

meters. The initial tests have been made without applying the magnetic field to get the average true heading of the path. However, the mag tests are conducted similar to the previous tests. i.e, the external magnetic field is applied only for the specified distance to observe the heading recovery after removing the external magnetic field. In this test, the magnetic field is applied, when the vehicle is about to take turn and removed when it is completed the turn, is given in Table.A.2, Appendix.A.5. This can be clearly observed in the Fig.2.22 and as expected, when the external magnetic field is removed from the vehicle, the IMU heading estimations follows true values. Therefore, the rejection of external disturbances for the given range can be easily avoided by fusing the vehicle kinematics information. The similar trends have been observed, when the vehicle is travelling in reverse direction in the same path. For brevity, the plots for all test conditions have not been shown in the paper. However, the RMS errors and test to test variation error for each test condition is given in Table.2.1.

Fig.2.21 provide reference curve for 90 deg east to north test condition by averaging the four similar mag-free test data sets and it is used for analyzing the mag-affected heading data.
From Fig.2.22 and 2.23 it is seen that, though the Norm of mag field cross the mag-free zone, the kinematic fusion estimates follow smooth transition and sudden changes have been eliminated. This shows the robustness of the fusion algorithm and ability to reject the mag-affected data.

Fig.2.24 shows the least square error deviation from true values along the path. While the vehicle is in Mag zone (from 7-22 m duration), the conventional EKF estimates suffer with magnetic disturbance and shows large deviations in heading estimates. However, the kinematics-fusion algorithm rejects error prone data from magnetometer and minimizes the deviations in heading estimates. This is quantified in Table.2.1, where RMS-errors for the given test condition improved from $6.0 to 1.9°$ without considering the test to test variation error of 0.7°.

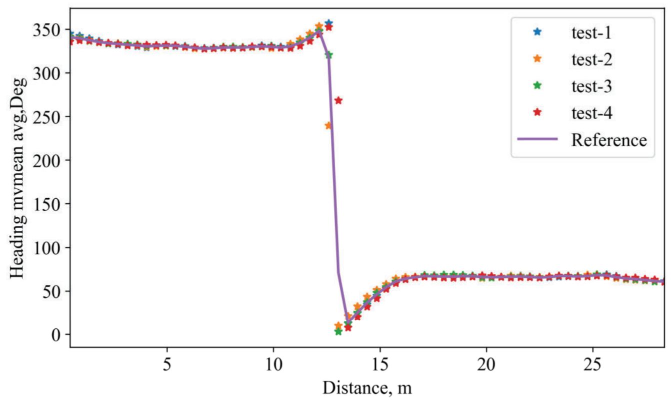


Figure 2.21: Reference heading for the 90 deg east to north navigation test.

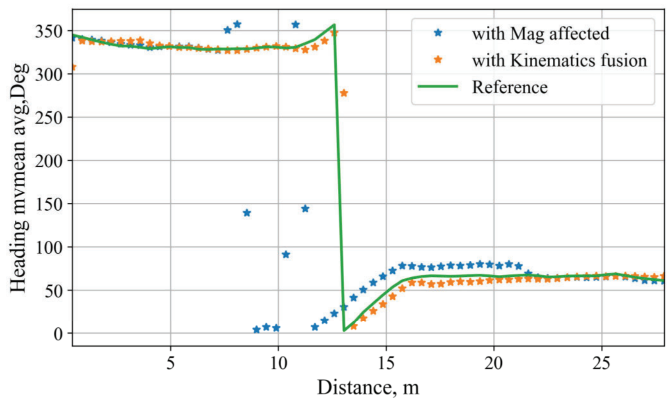


Figure 2.22: Comparison of vehicle heading with and without kinematics fusion algorithm for 90 deg east to north test condition.

2.4.5 Round-about turn test path Analysis

In these tests, the vehicle travels in straight line in the West direction and takes 180 deg round-about turn with 5m circle radius in the middle of the test path as shown in Fig.2.14. Further, the mag field is applied when the vehicle is at middle of the round-about circle and removed when the vehicle exit round-about circle is detailed in Appendix.A.5. in Table.A.2. This way, the performance of kinematics fusion algorithm and ability to differentiate mag affected data and avoid deviating from true path while vehicle is in typical round-about turns is studied. Fig.2.25 provides the reference or true heading for the given test condition by averaging the four mag free test data sets. This reference curve is used for making the error analysis with mag-affected tests. Further, the test to test variation error is provided in Table.2.1.

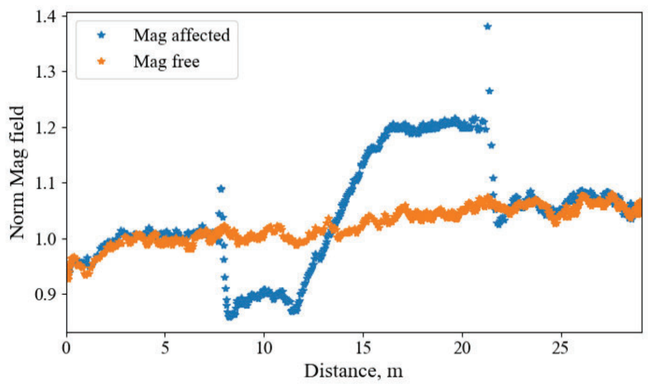


Figure 2.23: Comparison in Norm of Mag field for 90 deg east to north test condition.

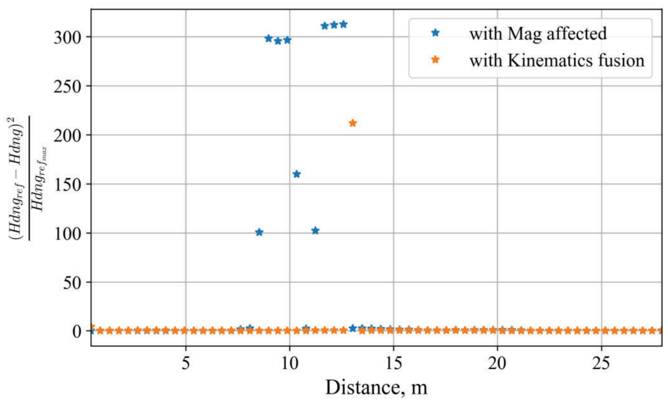


Figure 2.24: Least square error change along 90 deg east to north test path.

The comparison between reference heading (Fig.2.25) and mag effected heading is given in Fig.2.26. Form Fig.2.26 and 2.27, at the distance of about 14 m from starting point, it can be seen that the heading measurements deviates from its true values as the Norm of mag field deviates from its unit norm mag field. Though the variation in Norm mag field is varying considerably during Mag-zone, the Kinematics fusion algorithm relies on other parameters including yaw rate, rate of change of Norm mag field and Norm mag field sum to remove erroneous heading and make a smooth transition between Mag to Mag-free zone. Further, from Fig.2.28, the maximum least square error along the path can be observed when the Norm mag field is varying in abrupt manner. During this period, the kinematics fusion algorithm is robust and able to reject the mag affected data to keep the vehicle along the true path. This performance is quantified in terms of RMS error in heading estimations along the path, which is reduced from $1.9 to 0.3^\circ$ without considering the test to test variation error of 0.16° is given in Table. 2.1.

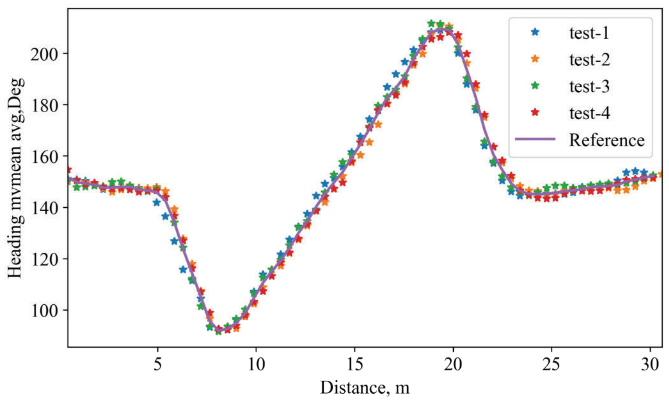


Figure 2.25: Reference heading for the 180 deg towards West test condition.

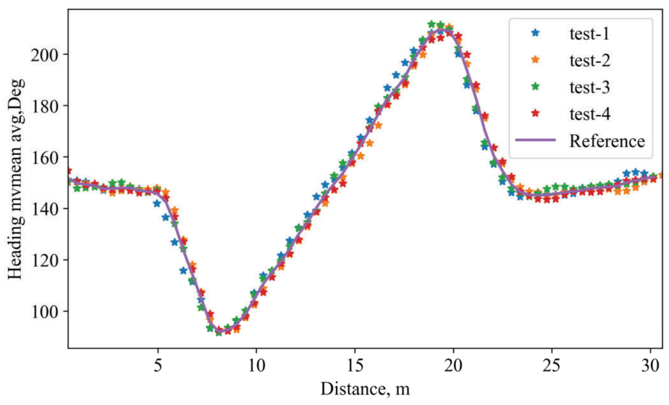


Figure 2.26: Comparison of vehicle heading with and without kinematics fusion algorithm for 180 deg towards West test condition.

2.4.6 Circle tests

The circular test has been made with a predefined path of 5 m circle radius and vehicle completes 3 circles in one test condition. As mentioned, mag tests are made in a strategic manner and here external magnetic field is applied only during 2nd circle maneuver. During, this Mag-zone (between 75-130 sec period), the sudden surge of external magnetic field causes the heading estimates to diverge from its true value as observed in Fig.2.29 and 2.30. There is a small difference shown in Figure 30 between reference heading and kinematic fusion heading in the mag affected zone due to the test to test variation and vehicle handling during the test. The total error in heading estimations from reference values has been quantified using RMS-errors and the kinematics-fusion algorithm reduces this error from $3.9 to 0.9°$ as given in Table.2.1.

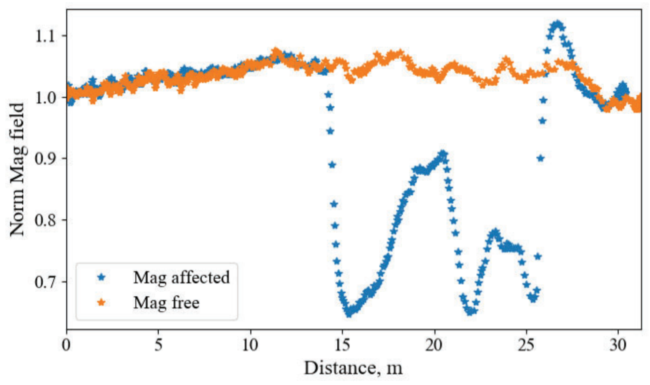


Figure 2.27: Comparison of Norm of Mag field for 180 deg towards West test condition.

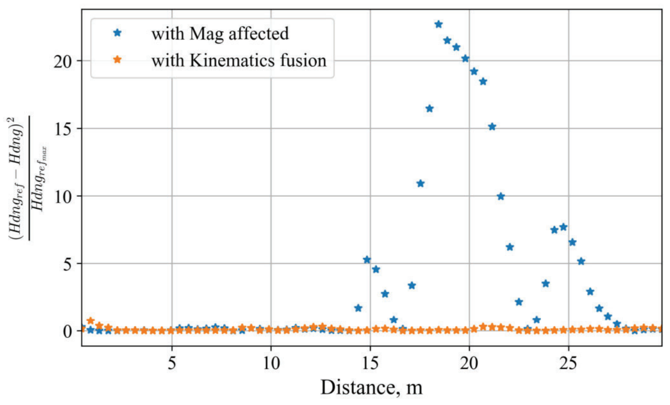


Figure 2.28: Least square error change along 180 deg towards West test path.

The true error will be lower as the kinematics-fusion includes test-test variation. From the results and discussion section, it can be clearly seen that, the vehicle kinematics fusion algorithm improved heading estimates while IMU is affected by external magnetic field disturbances. Further, RMS errors for each test condition quantifies the amount of improvement in heading estimates by using the kinematic fusion algorithm.

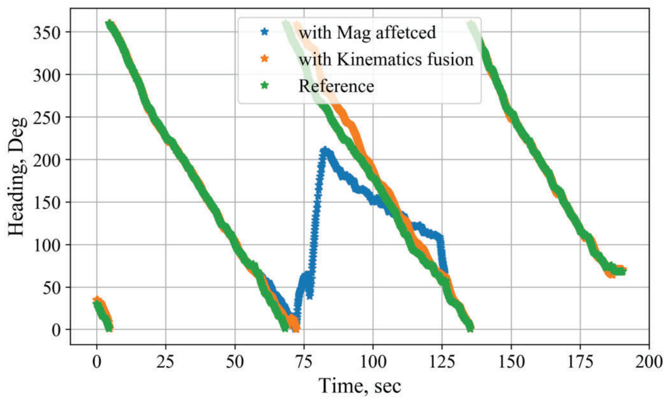


Figure 2.29: Comparison of vehicle heading with and without kinematics fusion algorithm for circular test condition.

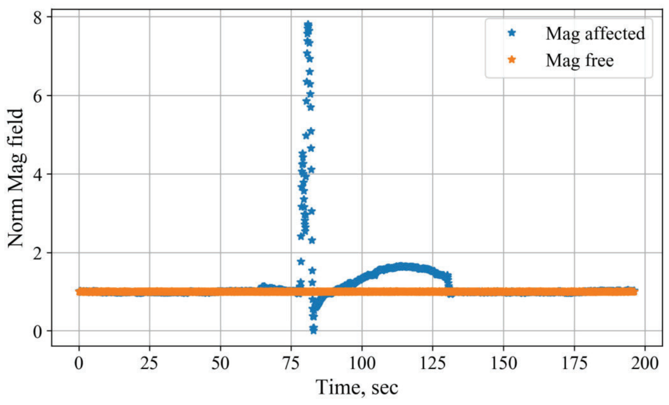


Figure 2.30: Comparison in Norm of Mag field for circular test condition.

2.4.6.1 Error Analysis

In this paper, the least square errors and RMS errors have been defined as follows:

$$Least\ Sq.\ Error = \frac{(Hdng_{ref} - Hdng)^2}{Hdng_{ref_{max}}}$$

$$RMS\ Errors = \sqrt{\frac{\sum_{i=1}^{n}(Hdng_{ref_i} - Hdng_i)^2}{n}}$$

Here, n indicates number of samples in the test data. The RMS error results for each test condition has been given in Table.2.1. The RMS error for the results with the kinematics fusion algorithm includes test to test variation but still it is seen that that on average the RMS error is reduced by more than 1.5° over the maneuvers.

Table 2.1
RMS errors in heading (in °) for different test conditions

Test Condition	Test-Test Variation RMS error	Without Kinematics Fusion Algorithm	With Kinematics Fusion Algorithm
St. line towards West	0.08	3.4	0.5
St. line towards East	0.04	1.3	0.3
90 Deg turn West to North	0.12	1.1	0.4
90 Deg turn South to East	0.2	2.3	0.6
90 Deg turn East to North	0.7	6.0	1.9
90 Deg turn South to West	0.12	1.5	0.4
5m circle roundabout towards West	0.16	1.9	0.3
5m circle roundabout towards East	0.8	1.7	0.9
7m circle roundabout towards West	0.2	1.5	0.8
7m circle roundabout towards West	0.9	2.8	1.8
circle test in clock-wise direction	1.05	3.9	0.9

Chapter 3

Non linear MPC algorithm development for unstructured environment with moving and stationary obstacle avoidance

3.1 Introduction

In recent years, the need for autonomous navigation has increased and world is tending towards making commercial and military vehicles into autonomous vehicles. This would be critical in off-road military utility applications, where the needy objects can be transported without human inference. However, in order to complete this task vehicle need to meet safety constraints at all times by accurately estimating the surrounding information and navigating through unexpected moving and stationary obstacles. This may be achievable in small robot on-road applications, where the priori information of obstacles are known [23]. However the recent advancement in embedded systems and performance in handling lager sets of information created an opportunity to develop autonomous navigation in passenger and larger vehicles too. Most of these vehicles autonomous navigation is still under critical review and it is especially true in case of high speed off-road application, where the priori information of obstacles and steep regions are unknown. Therefore, handling high speed off road conditions, which encounters sudden obstacles and steep regions requires thorough analysis on handling vehicle dynamics and thus making safe navigation. As of now, there exist some of the algorithms that can utilize the vehicle dynamics to navigate the vehicle safely, without collisions and yet maintain the high speed [24], [25]. However, these algorithms do not consider road grade and moving obstacles and requires potential research work to be made.

There are many obstacle avoidance algorithms available in the literature and they can be mainly divided into four categories including 1). Virtual potential and navigation function based methods [26] [27], 2). Graph-search based methods [23], [28], 3). Meta-heuristic-based methods [29] and 4). Optimization based methods [30, 31, 32, 33]. Each mentioned method might have served for the intended application. However, for the present off-road high speed application, the first two methods would nearly failed to handle dynamical safety requirements. However, the third method often suffers with computational burden and does not produce smooth commands for the navigation. However, the Optimization-based methods rely on rigorous mathematical formulations and offer systematic handling of vehicle dynamic safety constraints to ensure vehicle safety and yet generate smooth trajectories. Therefore, among the mentioned methods, optimization methods would most suits for the present application. The autonomous navigation can mainly be performed in two ways, firstly using preloaded map with offline planning and tracking the path using online feedback controller. Secondly, the optimum path and tracking would be made online by using current vehicle states and surrounding information. The later method be handled effectively by using Model predictive control (MPC) [34], [35] and best suits for the present application. In MPC, the optimal control problem (OCP) would be formulated through systematic mathematical representations to mimic future conditions and solved repeatedly over a finite control horizon. However, the controller would only consider initial part of the optimal solution for the implementation and thus recedes itself into optimum zone for the next iteration. Due to its ability to look into future and make decisions mimics human behavior in real time scenario. However, thorough analysis in developing vehicle model for state estimations and careful consideration of surrounding information needs to be made.

Previous studies have been made and succeeded by employing the MPC technique for obstacle avoidance in Autonomous ground vehicles (AGV) including [36, 37, 38, 39, 40, 41]. However, most of these studies have been restricted to on-road applications, where the surrounding information is structured in terms of road lanes, smooth and steady road grade, rules to follow traffic moments and no moving objects into your path. On the other hand, the present work considers developing AGVs in off-road unstructured environment with non-smooth road grades, including military applications. In this context, the 'unstructured environment' represents the path with no lanes, bumpy regions and no rules to follow traffic conditions. The objective of the present AGV is to navigate from its initial position to target position safely and as soon as possible. Therefore, by meeting the safety constraints, vehicle can travel at its maximum allowable speed and eliminate the constant speed condition. This reduces unnecessary deceleration in the navigation and reaches the target as fast as

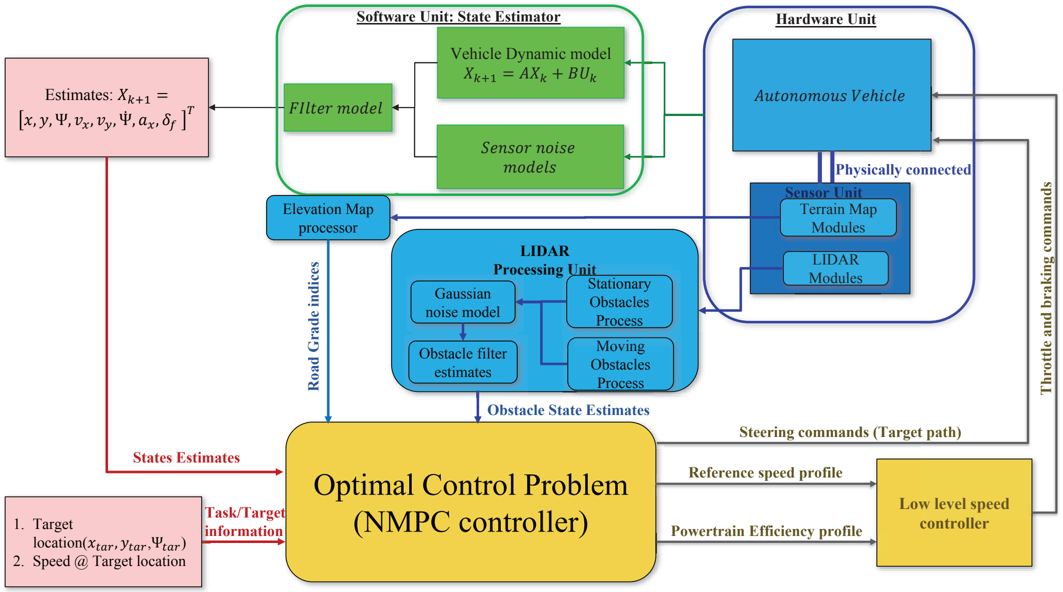
possible. While the vehicle is in travel, there exist both moving and stationary obstacles, whose position, speed, angle of shoot is not known a priori. However, these obstacles information can be estimated when they come into the range of planar light detection and ranging (LIDAR) sensors. In the present study, multiple LIDAR sensors are considered on the vehicle with different mounting positions to detect both stationary and moving obstacles in wide range view and fuse them together to create collision free path. Here, the obstacle view would still not cover backend of the vehicle and more details are explained in LIDAR process section. It is also assumed that, the limited range of road grade information is known from the map and it covers the vehicle front region as explained in LIDAR process.
The present work can mainly be distinguished in three ways from previous research studies and motivated to formulate a modified MPC problem. Firstly, handling of stationary obstacles with no-prior information using 'Box slope search' method. Unlike the on-road applications, where the target path is defined and can be perturbed for a brief period to avoid structured obstacles, in the present work a new algorithm is developed to process the stationary obstacle states in the vehicle view region and thus varying the target heading according to the surrounding conditions. Therefore, the target path is part of OCP formulation and can be deviated and optimized based on the obstacle position information. This formulation requires thorough analysis of each desired obstacle states with respect to vehicle states and making sure that the search logic is not cumbersome and can be easily implemented in real time. However, this can be achieved by dividing the vehicle view region into sub sections and avoiding the unnecessary calculations for the obstacles that are not obstructing the vehicle path and yet falls under the vehicle view region.
Secondly, the consideration of vehicle's dynamical safety constraints, while the vehicle is in high speed unstructured environment with no-prior information of obstacles. Here, the assumption of flat surface in unstructured environment is not a valid assumption and thus road grade information is considered in the safety constraint formulation. Prior algorithms considered vehicle dynamical safety constraints through excessive side slip including passenger cars on wet road or race cars [32]. However, these constraints are not adequate for the off-road heavy duty vehicles including military vehicles. These vehicles have high center of gravity (CoG) and it is critical to consider wheel lift-off condition as major dynamical constraint than wheel sideslip. Prior work has been made with wheel lift-off constraint to handle vehicle dynamical constraint in unstructured environment [25]. However, the study considered unstructured environment is flat thus omitted the effect of road grade on wheel lift off constraint. On the otherhand, due to road grade considerable load transfer in the vehicle longitudinal direction occurs and causes a shift in the vehicle CoG location. Therefore, it is critical to consider the effect of road grade in vehicle safety constraints to ensure all four wheels are on the ground.
Third novelty is related to the handling of moving obstacles and fusing its state estimations in the MPC formulation to avoid collision in real time. i.e., the algorithm

not only detects the moving obstacles but also includes them as state vectors in MPC system dynamics model to predict its states into the future and create collision free path. Several algorithms have been developed to detect moving obstacles [42],[43] in structured and small robot applications. Further, these studies are constrained to kinematic models and decoupled from moving obstacle states. [44] developed moving obstacle avoidance using forbidden velocity map within the defined dynamic window. However, the collision free path is not guaranteed and the adequate safety constraints and fusion of obstacle predictions with vehicle states may not be obtained.
Prior work has been made to avoid stationary obstacles in an unstructured environment using non-linear MPC algorithm [45], [46]. However, these studies considered constant longitudinal speed in the problem formulation and constrained the mobility performance. Further, the constant speed algorithm fails when it encounters bigger obstacles in the path and thus limit its performance to smaller obstacles. [25] extended the previous studies that consider optimal longitudinal speed and dynamical wheel-lift off constraints. The study also considered varying prediction horizon to achieve constant distance prediction. However, the study does not consider road grade and moving obstacles into the problem formulation. The omission of road grade in the study restricted wheel-lift off constraint to rear wheels alone. Further, it processes the LIDAR information at each iteration for the whole vehicle view region. It consumes considerable time and power for processing the LIDAR information and may not be feasible for the real time implementation.
The present work developed a new non-linear MPC problem formulation, that consider both stationary and moving obstacles with high speed navigation and yet meeting the vehicle dynamic safety constraints in an unstructured environment. The following major points can be highlighted from the present work
1) A new method called *ABD-JDN algorithm* has been developed for processing LIDAR information for stationary obstacles in the vehicle view region. This method divides the vehicle view region into sub sections and process each sub region in a strategic manner to find collision free path. Once the algorithm determines its path, it omits other sections for the obstacle search and thus avoid unnecessary processing in vehicle view region.
2) The inclusion of road grade into the dynamical safety constraints has been made. Therefore, the no-wheel lift-off constraints have been extended to all four wheels, as the road grade creates substantial longitudinal load transfer based on the direction of road grade. Further, the constraints have been implemented through both hard and soft constraints by using vehicle dynamics equations. Hard constraints are made, such that the vertical load on each wheel should be more than a specified minimum threshold load. In the present study, the minimum threshold load is assumed to be 1000 N. On the otherhand, the soft constraints are imposed to provide a smooth approach to the threshold load.
3) A new method is developed to process the moving obstacles that would estimate moving obstacle states including speed, position and angle of shoot. Based on the

angle of shoot, the algorithm decides, whether to consider the object as dangerous to do further processing. The algorithm fuses moving obstacle information into MPC problem formulation. A new MPC problem formulation has been developed by incorporating the moving obstacle states while keeping all previous conditions and constraints unchanged.
The present study considered vehicle parameters that mimic large, high speed military truck with significant vehicle dynamics and the acceleration,speed limits are taken from the previous studies made by [25]. However, as mentioned earlier the vehicle dynamical constraint is not restricted to rear wheels alone due to road grade consideration and all for wheels should meet the vehicle dynamic safety requirement. Further, the present work uses MPC frame work to formulate and solve the provided Optimal Control Problem (OCP).
The following assumptions have been made while developing the algorithm and these are explained in detail in the later Discussion section.
1. LIDAR sensor can detect all obstacles within the given range
2. The road grade is obtained through terrain maps and it is accurate for the given field range
3. Vehicle parameters are constant
4. Vehicle state estimations are free from noise
5. All moving obstacles can be detected and processed in the given moving obstacle LIDAR view region.
The MPC provide optimal solution for each prediction step in one iteration and in the present study, optimality refers optimal solution at each prediction step is based on the current information availability and not based on the information availability at all corresponding prediction steps. Further, the terms included in OCP makes it non-convex in nature and the global minima is not guaranteed. Therefore, in the present context, the optimal solution including optimal trajectory, optimal control inputs refers the local optimal solution calculated by the OCP solver.

3.1.1 Organization of work

The present work deals with stationary and moving obstacles along with road grade inclusion in the problem formulation. The rest of the paper is organised as follows; Section II provide methodology for stationary obstacles, in which the schematic of AGV and mathematical formulation of MPC algorithm for stationary obstacles along with road grade is furnished. Section III presents moving obstacle methodology, which incorporates mathematical formulation of moving obstacle states to existing stationary obstacle MPC formulation. Section IV provide the problem formulation on CasADi platform. Section V provide the simulation results for both moving and



Software Unit: State Estimator
Vehicle Dynamic model $X_{k+1} = AX_k + BU_k$
FIlter model
Sensor noise models
Estimates: $X_{k+1} = [x, y, \Psi, v_x, v_y, \dot{\Psi}, a_x, \delta_f]^T$
Hardware Unit
Autonomous Vehicle
Physically connected
Sensor Unit
Terrain Map Modules
LIDAR Modules
Elevation Map processor
LIDAR Processing Unit
Gaussian noise model
Stationary Obstacles Process
Obstacle filter estimates
Moving Obstacles Process
Road Grade indices
Obstacle State Estimates
Throttle and braking commands
Steering commands (Target path)
States Estimates
Optimal Control Problem (NMPC controller)
Reference speed profile
Powertrain Efficiency profile
Low level speed controller
Task/Target information
1. Target location($x_{tar}, y_{tar}, \Psi_{tar}$)
2. Speed @ Target location

Figure 3.1: Schematic of obstacle-grade avoidance layout using MPC algorithm.

stationary obstacle formulations and finally section VI makes the discussion on assumptions made in the present study and possible future work.

3.2 Methodology for stationary obstacles

The Fig. 3.1 shows the schematic of Non-linear MPC algorithm structure to avoid obstacles and steep road grade regions for AGV off-road application. The mathematical formulation of stationary obstacle avoidance serve as basis for moving obstacle avoidance by incorporating additional states and constraints in it. Therefore, the detailed explanation of cost function and constraints for the stationary obstacle avoidance is provided first, and then the moving obstacle process would be incorporated in later sections.
This section is mainly divided into three parts, in which part-1 provides overview of MPC algorithm and its structure, and part-2 presents obstacle-road grade process, also called ABD-JDN algorithm and part-3 provide mathematical formulation of OCP for stationary obstacle and road grade avoidance.

3.2.1 Part-1: Overview of MPC algorithm

The basic level overview of algorithm and its inputs,outputs,plant model and algorithm objectives are briefed in this section.

3.2.1.1 MPC algorithm Structure

As mentioned, MPC is most suitable for the autonomous navigation as it looks into the future for the given horizon to simulate real time human driving conditions [35] and make appropriate decisions to maneuver safely. The MPC uses system dynamic model and surrounding information to formulate the optimal control problem systematically and provide the optimal solution at each prediction step in one iteration. However, the algorithm only implements initial prediction step solution called receding horizon control on the plant and a new OCP would be formulated over the next iteration. This enables handling of real-time optimization with hard constraints on the plant model [34].However, the performance of the MPC algorithm mainly depends on vehicle plant model, which needs to be modelled carefully to mimic the actual vehicle dynamics.

3.2.1.2 Vehicle plant model

In the present work, the vehicle is modeled with 8 DoF [47] and it refers a 101 DoF multibody dynamics model of a truck. The powertrain dynamics are modeled using [48], [49] and nonlinear tire model has been modeled using [50]. For brevity, the powertrain model has not been explained here and above mentioned references provide more details for it. The vehicle parameters along with MPC tuning parameters are provided in Table. 3.2. The model mimics the real vehicle system and it is referred as vehicle plant model in the rest of the paperwork.i.e., the model can be replaced with actual vehicle with appropriate inputs and outputs from external sensors and actuators, the actual vehicle should be able to avoid stationary obstacles along with steep regions.

3.2.1.3 Algorithm objective

The main goal of present algorithm is to move AGV from its current position to target position as soon as possible in an unstructured environment by meeting all safety constraints. Here, the safety constraints include avoiding the stationary obstacles, steep regions and preventing vehicle from rollover. Further, as mentioned the unstructured environment refers roads without lanes and traffic rules, no priory information of obstacles and steep regions. Therefore, the LIDAR modules be used for obtaining the road and obstacle field conditions and the algorithm should be able process its information and command the AGV towards target location as quickly as possible by meeting the above safety constraints.

3.2.1.4 Algorithm inputs and outputs

The algorithm mainly requires three types of inputs that would provide sufficient information to reach target location safely. The three inputs include current information, target information and environment information is show in Fig. 3.1. In the present study, the current information includes the current state estimates, which can either be measured or estimated using the state estimator. In practice, it is not possible to measure all the states using sensors and an estimator is necessary to provide current vehicle information.
The target information includes target position, desired heading and speed at target location. Further, the environment information includes the obstacle location and road grade indices and it is obtained through LIDAR modules and terrain map process respectively. More details on obstacle and road grade processing and avoidance is explained in part-2 of ABD-JDN algorithm section given below.
The outputs from the algorithm includes steering angle vehicle speed commands, that would avoid all obstacles and reaches target location as soon as possible. Here, though the algorithm produces reference vehicle trajectory, the present work has not developed low level speed controller to mimic the real world and it is left for the future work.

3.2.2 Part-2: ABD-JDN algorithm

This section deals with environment data process and identifies obstacle-grade free regions in the given path. In the present work, the ABD-JDN algorithm for obstacle-grade processing is made in two stages. In the first stage, the road is assumed to be flat and the algorithm is developed for avoiding the stationary obstacles alone. In stage two, road slope is incorporated and the algorithm is extended to avoid both obstacles and road grade in the defined LIDAR view region. Further, the LIDAR view is defined based on the vision sensor maximum range and its ability to detect obstacles with less noise is shown in Fig. 3.1. For instance, the Leddar VU8 sensor [51] provide eight segments of sensor view, that are used for identifying obstacle distance and position. Each segment can be extended for upto 150 m range with 20° horizontal spread angle. However, for the better accuracy of sensor measurements, it is advisable to restricts its range to 70-100 m. Therefore, the present algorithm is developed by considering the factors of real world sensor range, update rates and its operating principles.
The following sub sections provide complete details on ABD-JDN algorithm with appropriate definitions and figures in it.

3.2.2.1 LIDAR View definition

This is some times referred as Vehicle view or LIDAR-1 view region for stationary obstacles, in the rest of the paperwork. In the present work, the function for obstacle and road grade detection in the vehicle view region is developed through set of rectangular boxes and each box can closely mimic Leddar VU8 sensor visibility [51]. The coordinates for each box is made with respect to the vehicle current position and heading is shown in Fig. 3.2. Based on the current heading, there are 7 boxes made for both left and right sides and each box has longitudinal length (height) of 75 m, and lateral (width) length of 3.5 m.
The right boxes are indexed with positive numbers and vice versa. i.e., the straight path along the vehicle current heading is defined with (1,-1) boxes, which covers the area of 75x7 m^2 is shown in Fig. 3.3(a). This way, a pair of two boxes has been taken for search algorithm to make sure the area encloses the vehicle dimension and avoid obstacles in both lateral and longitudinal directions. Therefore, the LIDAR view is mainly divided into 3 sections on each side of the vehicle along with straight path shown in Table. 3.1.

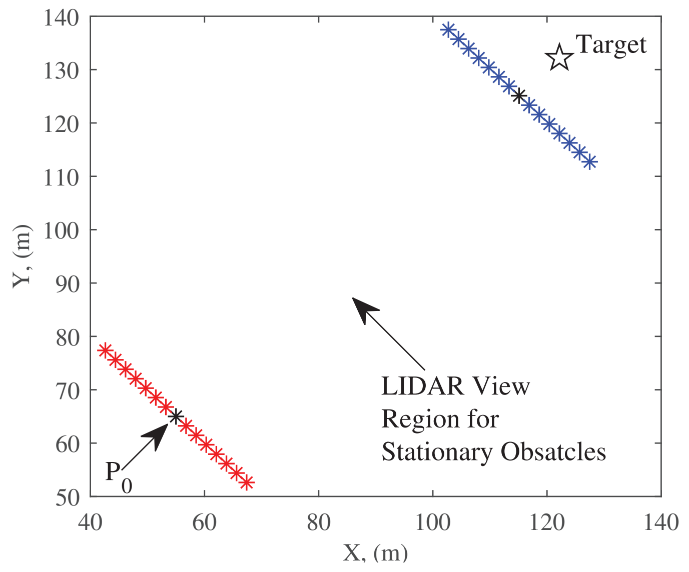


Figure 3.2: LIDAR view region for obstacle-grade analysis.

Table 3.1
LIDAR view sub sections based on box coordinate system

Box region	Path	Heading change
Box region: [-1, 1]	Straight path	No-change in heading
Box region: [2, 3]	Right-1 path	Current heading – pi/8
Box region: [-2, -3]	Left-1 path	Current heading + pi/8
Box region: [4, 5]	Right-2 path	Current heading – (2*pi/8)
Box region: [-4, -5]	Left-2 path	Current heading + (2*pi/8)
Box region: [6, 7]	Right-3 path	Current heading – (3*pi/8)
Box region: [-6, -7]	Left-3 path	Current heading + (3*pi/8)

3.2.2.2 Obstacle detection logic

The ABD-JDN algorithm mainly has two parts including 1). Obstacle detection process 2.) Obstacle avoidance process. The obstacles in this context can include road grade and moving obstacles too. However, the algorithm varies slightly for each scenario and details for each scenario is explained in the following sections.
As mentioned, the first part of ABD-JDN algorithm includes obstacle detection and followed by avoidance process. Therefore, this section provide complete details on obstacle detection process. The obstacle detection process works based on the vehicle current heading, slope of line joining between two points and corner points. As

mentioned, the box co-ordinates (or corner points) are made parallel to the current vehicle heading and distributed along lateral direction with a width of 3.5 m in vehicle left and right directions is shown in Fig. 3.3(a). Further, starting with left-bottom corner point, four corner points would be connected in clock-wise direction to create a rectangular box for obstacle search region. From Fig. 3.3(b), it can be seen that, the left box in straight path consists of four corner points starting from its left bottom corner point 1 to 4 in clock-wise direction. Similarly, the starting location for right box in straight path would start from left-bottom corner point which becomes vehicle current position. i.e, corner point 4 for left box becomes starting point 1 for the right box in straight path. Therefore for the obstacle detection method, the corner points along with slopes of lines joining corner points in clock-wise direction acts as limits for detecting and processing the obstacle states in the LIDAR view region. This process is done in two steps in the following manner,

I). The obstacle (x_{om}, y_{om}) position should be within the limits of diagonal corner points of box. For instance, if an obstacle is identified in left box in straight section the x co-ordinates of obstacle $((x_{om})$ is compared with x co-ordinates of box diagonal corner points 1 and 3, and y co-ordinates of obstacle $((y_{om})$ is compared with y co-ordinates of box diagonal corner points 2 and 4 to make sure it is in the box limits.

II). The slopes of four lines, which are obtained by joining four box corner points with obstacle point are compared with slopes of lines, that are made by joining box corners in clockwise direction.

Therefore for the given box-region, the above two conditions need to satisfy in order to confirm the obstacle detection in the corresponding box region. However, it is important to note that the logic slightly changes its signs as the heading of the vehicle varies from 0 to 360°.

3.2.2.3 LIDAR data storage process

Once the obstacle is found in LIDAR view region, the ABD-JDN algorithm stores its corresponding detected obstacle states in a temporary memory to provide environment information to the MPC controller. However, due to the limited space and computational burden, the temporary memory can store upto 76 obstacles information. Further, the logic continuously updates the detected obstacles, that are within the 125 m perimeter from the current vehicle position and replaces the far distance obstacles information. This way, the LIDAR data process provide appropriate environment information to navigate the vehicle in smoother and controlled manner to ABD-JDN algorithm and it is explained in the following sections.

3.2.2.4 ABD-JDN Algorithm: for stationary obstacle avoidance process

The second part of the ABD-JDN algorithm includes the obstacle avoidance process and it depends on the stored obstacles in its LIDAR view region. The following section explains about obstacle avoidance process in detail without considering the road grade information in it.
The schematic of Obstacle avoidance process for a typical situation is given Fig. 3.3. The process outputs target heading and target velocity based on the obstacle position from vehicle current location and is explained in following sections.
As mentioned, the obstacle detect function makes use of the current vehicle position and heading for creating box coordinates to process the LIDAR module data. Here, assume that the each LIDAR unit can detect objects in 2 boxes with 5 m width and 75 m longitudinal height respectively. From the Fig. 3.3(a), the obstacle avoidance process first searches in straight 2 boxes [1,-1] and if there are no obstacles in this region, the vehicle tries to move towards target point without any change in its target co-ordinates and travel with maximum speed. However, If there are any obstacles found in either of 2 straight boxes, the algorithm look for the obstacle information from its right 2 boxes [2,3] and if there are no obstacles in this region, the vehicle takes slight right turn from its current heading value is shown in Fig. 3.3(b). Further, if the right 2 boxes [2,3] find any obstacle, the algorithm look for the left 2 boxes [-2,-3] and if it finds obstacle free region, it navigates the vehicle towards left direction is shown in Fig. 3.3(c).
This logic repeats until it reaches the extreme left 2 boxes [-6,-7], which indicates the obstruction of entire LIDAR view region is shown in Fig. 3.3(d). If all the boxes or entire LIDAR view region is obstructed by an obstacle, the vehicle tries to move extreme right direction. However, it is important to note that, the vehicle target speed be changed based on the obstacle distance from the vehicle current position in straight section. i.e., if the obstacle is found in straight section, the vehicle tries to slow down first and then decides to go either right or left based on further search in the logic. In the present logic, the vehicle speed decreases linearly from 29 to 5 m/s as the obstacle distance changes from 75 to 5 m from the vehicle current location. The Pseudo code on above explained logic is given Appendix A.

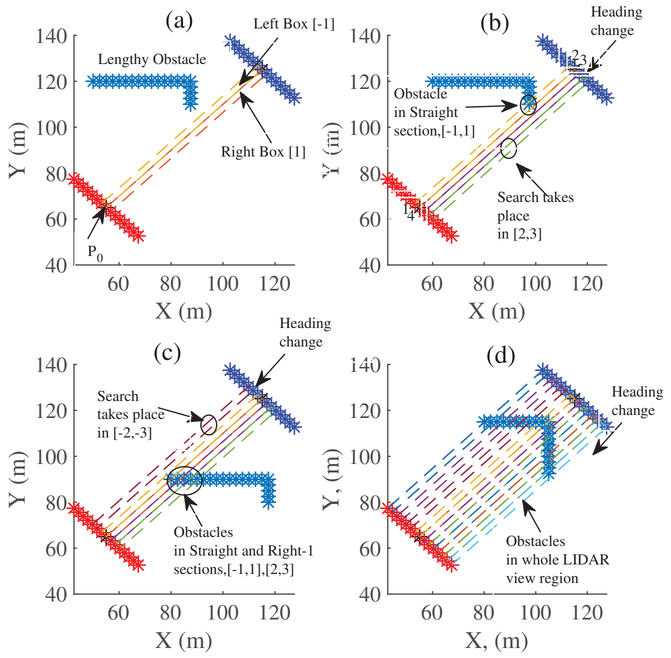


Figure 3.3: Schematic of stationary obstacle avoidance process.

3.2.2.5 ABD-JDN algorithm: for combined obstacle and road-grade avoidance process

As mentioned, this function outputs similar to the obstacle-avoidance process except that, this logic includes road-grade information too. Therefore, the combined obstacle and road-grade avoidance process provide the heading and velocity that would avoid collision, based on the grade indices, obstacle detections and vehicle current position.

Road-grade indices definition:

In general, the highway and city commuting road grades range between 0-6.8° and therefore, in the present work, it is assumed that the road grade more than 13.8° is considered an obstacle and below 6.8° is considered as flat road is shown in Fig.3.7. Further, it is difficult to process the road grade information for both lateral and longitudinal directions at each point in the LIDAR view region. The LIDAR view region approximately covers $75m$ in longitudinal direction and $5m$ in lateral direction. Therefore, the terrain map processing in the present made assumptions to process grade information to the ABD-JDN algorithm. The road grade is calculated at each $7m$ longitudinal distance for upto $75m$ in the given current vehicle direction and looks for the obstacle detection in the calculated region. If the road grade at all points meet obstacle free region, the algorithm calculates grade with the step size of $1m$ distance for upto $7m$ and averages for obtaining the grade in longitudinal direction (θ_x). This is because, it is important to look for potential obstacles due to grade and then calculate the grade that is near to the vehicle current position. This makes, vehicle to see look ahead and make accurate decision on grade avoidance. Further, the lateral grade (θ_y) be calculated by averaging the grades with step size of $1m$ in lateral direction at a longitudinal distance of 2.5m. The higher layer steps involved in algorithm are explained as follows

Step 1

When the algorithm finds no obstacle in straight path and having steep road-grade indices, the logic makes decisions using *Straight-path search* method.

Step 2

If there are obstacles in straight-path and no obstacle in right-1 path, the logic uses *Right-side search* method to navigate vehicle in obstacle free low-grade indices region.

Step 3

Further, if there exist obstacles in both straight and right-1 boxes, the logic uses *Left-side search* method to navigate the vehicle.

Step 4

The step-2 and step-3 are repeated until the logic reaches left-3 path. i.e., if the straight, right-1 and left-1 paths have obstacles, the logic looks for the right-2 path and applies *Right-side search* logic to navigate the vehicle. Further, if the right-2 path have obstacles, the algorithm makes search in left-2 path. In this alternative way, the logic repeats until it reaches maximum available paths (3 paths on each side) and once it completes all the paths and confirmed that the whole LIDAR view region is obstructed by obstacles, the vehicle maneuvers to extreme right direction is shown in Fig. 3.3(d). The lower layer logic sections including Straight-path search, Right-side search and Left-side search is provided in the following sections.

Straight-path search

Step 1

Similar to previous obstacle-avoidance logic, this logic searches for any obstacles in the straight-path. If there are no obstacles in this region, it looks for the bumps/road-grade information. The road-grade is explained in previous section. As mentioned, the logic treats road-grade as an obstacle, when its grade region is more than 13.8°. Therefore, the logic initially searches for any obstacles including grade regions $> 13.8°$ in the straight section and if there are no obstacles found, it further looks for the minimum road-grade indices explained in the following steps.

Step 2

once the straight path is free from obstacles (including road-grade>13.8°), the algorithm looks for the road-grade information in the straight section and if the road-grade indices is less than 6.8°, the vehicle does not change its direction and travels with top speed towards target location.

Step 3

However, if the road grade indices is more than 6.8 and less than 13.8°, the algorithm looks for the right-1 path (Box 2,3) boxes. i.e., if the right-1 path is free from obstacles (including grade regions $> 13.8°$) and grade indices is less than 6.8°, the vehicle algorithm outputs with heading that make vehicle to take right turn. However, if there is any obstacle found in right-1 path, the vehicle travels in straight path with reduced speed based on grade indices. Here, vehicle speed varies linearly between 29 to 2.5 m/s as the road indices changes from 6.8 to 13.8°.

Step 4

If the road-grade indices is more than 6.8 and less than 13.8° in both straight and right paths without any obstacles, the algorithm looks for the left-1 path (Box -2,-3) boxes. i.e., if the left-1 path is free from obstacles (including grade regions >13.8°) and grade indices is less than 6.8°, the vehicle algorithm outputs with heading that make vehicle to take left turn. However, if there is any obstacle found in left-1 path, the vehicle travels in straight path with reduced speed based on grade indices. However, if all of 3 paths including straight, right-1 and left-1 paths are having grade indices between 6.8 and 13.8° without obstacles, the algorithm outputs with heading and speed that make vehicle to turn in low grade-indices path. As mentioned, the algorithm also outputs the road-grade indices to main controller to meet the vehicle lift-off constraint on each tire when vehicle travels on grading location.

Right-side search

This search is made, when the vehicle identifies obstacle in the straight section and no obstacle in the right-1 path. As mentioned, in this logic any road-grade region $>$ 13.8 is also considered as an obstacle. The right-side search method can be explained as follows

Step 1

As mentioned, when there are obstacles in straight path vehicle initially looks for the right-1 path. If there are no obstacles in right-1 path, it looks for the road-grade information from right-1 path and if it is below 6.8°, the vehicle take diversion towards right-1 path. However, if both straight and right-1 paths have obstacles, then the logic makes search in left-1 path is explained in Left-side search section.

Step 2

on the other hand, if the grade indices is between 6.8 to 13.8° without any obstacles in right-1 path, the algorithm look for the right-2 path (box 4,5) boxes. i.e., if the right-2 path is free from obstacles (including grade regions $>$ 13.8°) and grade indices is less than 6.8°, the vehicle algorithm outputs with heading that make vehicle to take right-2 turn. However, if there is any obstacle found in right-2 path, the vehicle travels in right-1 path with reduced speed based on grade indices is explained in previous steps.

Step 3

If the road-grade indices is more than 6.8 and less than 13.8° in both right-1 and right-2 paths without any obstacles, the algorithm looks for the right-3 path (Box 6,7) . i.e., if the right-3 path is free from obstacles (including grade regions>13.8°) and grade indices is less than 6.8°, the vehicle algorithm outputs with heading that make vehicle to take right-3 turn. However, if there is any obstacle found in right-3 path, the vehicle travels in the direction of minimum grade indices between right-1 and right-2 paths with reduced speed. However, if all of 3 paths including right-1, right-2 and right-3 paths are having grade indices between 6.8 and 13.8° without obstacles, the algorithm outputs with heading and speed that make vehicle to turn in low grade-indices path. Here, the algorithm checks for maximum paths available for making obstacle-grade search and if it reaches maximum limit (3 paths on right side), the vehicle stops search and makes the way into the minimum grade indices path. The present study considered maximum of 3 paths on each side f vehicle. However, the algorithm can be easily be extended to more number of paths to increase the obstacle-grade search region and this would eventually increases the computational burden on MPC formulation.

Left-side search

As mentioned, this search is made, when the vehicle identifies obstacles in both straight and right-1 paths and no obstacle in the left-1 path. The logic for left-side search is similar to the right-side search and maneuver the vehicle in obstacle free low grade-indices path. The detailed flow chart on the above ABD-JDN algorithm for Obstacle grade avoidance process is provided in Appendix .B.3.

3.2.3 Part-3: Optimal control problem formulation using MPC for Stationary Obstacle avoidance

$$\min_{\mathcal{Z},\mathcal{U},O_d} \quad J = \mathcal{T}[\mathcal{Z},\mathcal{U}] + \int_0^{T_p} [\mathcal{Z}(t),\mathcal{U}(t),O_d(t)]dt \tag{3.1}$$

s.t.

$$\dot{\mathcal{Z}}(t) = \mathcal{D}[\mathcal{Z}(t),\mathcal{U}(t),\theta] \tag{3.2}$$

$$\mathcal{Z}_{min}(t) \leq \mathcal{Z}(t) \leq \mathcal{Z}_{max}(t) \tag{3.3}$$

$$\mathcal{U}_{min}(t) \leq \mathcal{U}(t) \leq \mathcal{U}_{max}(t) \tag{3.4}$$

$$\mathcal{S}[\mathcal{Z}(t),\theta] \leq 0 \tag{3.5}$$

$$\mathcal{O}_{St}[\mathcal{Z}(t),O_d^j(t)] \leq 0 \tag{3.6}$$

$$T_p \in [T_{p_{max}},T_{p_{min}}], \mathcal{L}[\mathcal{Z},L_d,dt] \leq 0 \tag{3.7}$$

$$\mathcal{E}[\mathcal{Z}(t),\mathcal{Z}_{target}] \leq 0 \tag{3.8}$$

The optimal control problem can be formulated using the equations (3.1)-(3.8) and the detailed explanation on each constraint and cost function have been made in the following sections,

3.2.3.1 Equation (3.2): Dynamic model constraint

In the present work, the vehicle has been modeled with 8 states including $[x, y, \psi, v_x, v_y, \dot{\psi}, a_x, \delta_f]$ and 2 control inputs J_f, γ_f. Here, x, y represents the vehicle position with respect the center of gravity location and ψ represents the vehicle yaw or heading angle. Further, v_x,v_y indicates vehicle longitudinal and lateral velocity respectively. $\dot{\psi}$, is yaw rate in rad/s, a_x is longitudinal acceleration and δ_f is steering angle.Further, the control inputs for the system model are jerk (J_f) in m/s^3 and steering rate (γ_f) in rad/s. The ODE model to represent vehicle dynamics can be written in state space form as follows,

$$\dot{z} = A(z) + Bu \tag{3.9}$$

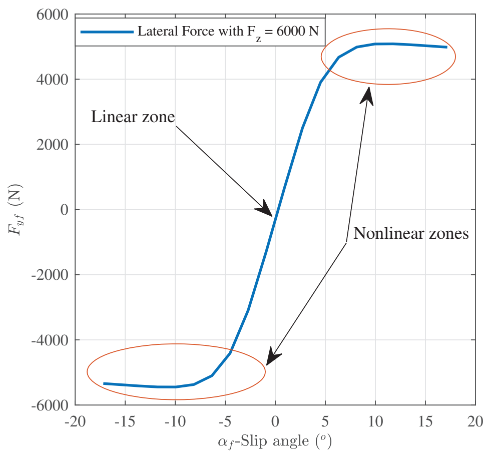


Figure 3.4: Pacejka magic formula for estimating lateral tire forces

Where,

$$A(z) = \begin{bmatrix} v_x \cos\psi - (v_y + l_f\dot{\psi})\sin\psi \\ v_x \sin\psi + (v_y + l_f\dot{\psi})\cos\psi \\ \dot{\psi} + (v_x/l)\tan\delta_f \\ a_x \\ (F_{yf} + F_{yr})/M_{total} - v_x\dot{\psi} \\ (F_{yf}l_f - F_{yr}l_r)/I_z \\ 0 \\ 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, vehicle parameters including l_f,l_r are called distance from front and rear wheel axle to vehicle C.G. location in m; M_{total} is total mass of the vehicle in kg and I_z is the moment of inertia about $z-axis$ in $kg-m^2$. For most of on-road low speed applications, the tire slip angles are low and they fall in linear zone and it is sufficient to use linear tire model for developing lateral controller. Therefore, the vehicle lateral states including y,ψ can be estimated using linear portion of Fig.3.4 by having constant cornering stiffness coefficient and vehicle parameters [52]. However, for high speed, off road applications the tire slip angles are large and linear portion of tire model is no longer valid. Therefore, for the present study a nonlinear tire model is required to estimate lateral forces and thus accurate vehicle lateral states is given in (3.9). Here, F_{yf},F_{yr} are called front and rear lateral tire forces in N respectively. These vehicle lateral tire forces are estimated using Pure-slip Pacejka-magic formula [50] with the function of vertical load F_z and slip angles α_* is shown in Fig. 3.4. The simple Pacejka tire model estimates lateral forces as follows,

$$F_{y*} = D_* \sin(C_* \arctan(X_* - E_*X_* + E_* \arctan(X_*))) \tag{3.10}$$

in the above equation $_*$ can be assigned to f, r with f being front and r being the rear tire respectively. Further, $X_* = B_*\alpha_*$ and B_*, C_*, D_*, E_* are function of vertical load F_z. The front and rear slip angles can be defined in terms of vehicle states and control inputs as follows

$$\alpha_f = \delta_f - \arctan(v_y/v_x) - (l_f\dot{\psi})/v_x \tag{3.11}$$

$$\alpha_r = -\arctan(v_y/v_x) + (l_r\dot{\psi})v_x \tag{3.12}$$

Here, α_f should be in deg and vertical load F_z should be in kN for estimating the lateral forces using Pacejka model. The vertical load on each axle for a single track bi-cycle vehicle model, without considering the longitudinal and lateral load transfer on a flat road can be written is as follows,

$$F_{z,f0} = (M_s l_r/l + M_{u,f})g \tag{3.13}$$

$$F_{z,r0} = (M_s l_f/l + M_{u,r})g \tag{3.14}$$

Where, M_s is the sprung mass, $M_{u,f}$ and $M_{u,r}$ are unsprung mass at front and rear side respectively. Further, the sprung and unsprung mass are estimated using [53], and the equations for calculating these parametres are,

$$M_{total} = M_s + M_u \tag{3.15}$$

$$M_u = 0.1412 * M_{total} \tag{3.16}$$

$$Mu_f = Mu_r = M_u/2 \tag{3.17}$$

However, during high speed and steep road conditions, considerable longitudinal load transfer occur and it is important to account for these changes in vertical load, such that the tire lateral forces in Pacejka model would be estimated accurately [50]. Therefore, the following terms would be added to account for longitudinal load transfer in vertical load calculations,

$$F_{z,f} = F_{z,f0} - \mu_{z,x}(a_x - v_y\dot{\psi}) - F_{gradeX} \tag{3.18}$$

$$F_{z,r} = F_{z,r0} + \mu_{z,x}(a_x - v_y\dot{\psi}) + F_{gradeX} \tag{3.19}$$

$$F_{gradeX} = (M_{total} g h \theta_x)/l \tag{3.20}$$

Here, $\mu_{z,x}$ refers to the longitudinal load transfer co-efficient for estimating vertical load shift due to acceleration and deceleration of vehicle [25]. The parameter F_{gradeX} is force due to road grade in longitudinal direction, N; h is the height of C.G. from ground in m; θ_x is the road grade in longitudinal direction, rad. From the Equations (3.18) and (3.19), it can be clearly observed that the positive road grade transfer longitudinal load onto rear wheels and negative road grade transfer load onto front wheels. However, for a single track bi-cycle model, it is assumed that left and right wheels lumped together into one wheel for both front and rear axles. Therefore, the following assumptions are made while calculating lateral tire forces [25],

Assumption-1

$$\alpha_{*,left} \approx \alpha_{*,right} \triangleq \alpha_{*} \tag{3.21}$$

As mentioned, $*$ represents for both $front$ and $rear$ and the Equation (3.21) implies the left and right slip angles are assumed to be same for calculating lateral tire forces.

Assumption-2

$$\digamma(F_a, *) + \digamma(F_b, *) \approx \digamma(F_a + F_b, *) \tag{3.22}$$

from (3.22), the tire lateral force is approximately in linear function with tire vertical load and thus eliminating the effect of lateral load transfer in tire lateral force calculations.
The detailed explanation on tire forces including lateral and longitudinal load transfers is given in [25] and briefly explained in vehicle wheel lift-off or dynamical safety constraint section.

3.2.3.2 Equations (3.3), (3.4):State and control constraints

As mentioned, the optimal solution generates control commands in terms of steering rate γ_f and jerk J_f. Here, steering rate command is used for lateral control and Jerk is used for longitudinal control respectively. These control commands ensure smooth vehicle operation compared to steering angle and speed as control inputs. Due to vehicle mechanical linkage limits on steering system, the steering angle and steering rate are restricted to between constant bounds and it it written as follows,

$$\delta_{f_{min}} \leq \delta_f(t) \leq \delta_{f_{max}} \tag{3.23}$$

$$\gamma_{f_{min}} \leq \gamma_x(t) \leq \gamma_{f_{max}} \tag{3.24}$$

On the otherhand, the acceleration is restricted based on the full load operation of power train and brake dynamics [25]. Here, N represents the number of predictions made during one MPC sampling period or iteration. Therefore, all the state and control command limits need to be satisfied for all vehicle future predictions in the given problem formulation.

$$a_{x_{min}}[v_x(t)] \leq a_x(t) \leq a_{x_{max}}[v_x(t)] \tag{3.25}$$

Therefore, the acceleration and braking bounds are function of vehicle speed and complete details on acceleration limits can be found in [25]. For brevity, the polynomial fit equations for acceleration and braking limits is provided through below equations (3.26), (3.27).

$$a_{x_{max}}[v_x] = c_1 v_x^3 + c_2 v_x^2 + c_3 v_x + c_4 \tag{3.26}$$

$$a_{x_{min}}[v_x] = c_5 v_x^3 + c_6 v_x^2 + c_7 v_x + c_8 \tag{3.27}$$

Further, based on vehicle powertrain and obstacle field density, the constant upper and lower bounds are imposed on vehicle longitudinal speed, Jerk and the bounds on vehicle yaw ψ is restricted between $0 - 359°$ to ensure vehicle navigation in all 4 quadrants is given in equations (3.28),(3.30).In addition, due to the mechanical limits of steering angle and vehicle dynamical safety constraints on each tire ensures implicit constraints on yaw rate $\dot{\psi}$ and lateral speed v_y and it is discussed in the next section.

$$v_{x_{min}} \leq v_x(t) \leq v_{x_{max}} \tag{3.28}$$

$$\psi_{min} \leq \psi(t) \leq \psi_{max} \tag{3.29}$$

$$J_{f_{min}} \leq J_f(t) \leq J_{f_{max}} \tag{3.30}$$

3.2.3.3 Equation (3.5):vehicle dynamical safety constraint

The vehicle safety for rollover prevention has been confirmed with the load on each tire should be minimum of $995N$. This constraint has to be satisfied at all times including steep roads, variable speed and steep cornering conditions. As mentioned, for variable speed and steep road conditions, considerable longitudinal load transfer takes place and it is important to account for longitudinal load transfer to ensure vehicle safety. Previous studies [32],implemented this constraint through steering angle limits or lateral acceleration limits [54] and omitted the longitudinal load transfer effects. Further, the prior work from [25], included longitudinal load transfer effects but restricted the vehicle safety constraint to rear wheels alone as it has assumed the flat road condition in the problem formulation. However, the inclusion of road grade in the current problem formulation creates substantial load transfer from front wheels during positive road-grade operation. Therefore, it is required to apply minimum vertical load constraint on all 4 wheels to make sure vehicle stability on the ground. Vehicle safety constraints for rollover prevention has been made through following equations,

$$F_{z,f-left} = 0.5.F_{z,f} - \Delta F_{z,yf} \tag{3.31}$$

$$F_{z,f-right} = 0.5.F_{z,f} + \Delta F_{z,yf} \tag{3.32}$$

$$F_{z,r-left} = 0.5.F_{z,r} - \Delta F_{z,yr} \tag{3.33}$$

$$F_{z,r-right} = 0.5.F_{z,r} + \Delta F_{z,yr} \tag{3.34}$$

The above equations from (3.31)-(3.34) estimate the load on each tire by considering the effects of grade, lateral and longitudinal load transfers [25]. Therefore, these estimated loads on each tire should meet the minimum specified vertical load to prevent the vehicle from rollover. Here, $F_{z,f}, F_{z,r}$ accounts for longitudinal load transfer for front and rear axles and are detailed in Equations 3.18,3.19. The $\Delta F_{z,yf}$ is for front axle lateral load transfer and $\Delta F_{z,yr}$ is for rear axle lateral load transfer. Further, these load transfers can be approximated through set of simulations including the effects from variable speed and road grade as follows,

$$\Delta F_{z,yf} = \mu_{z,yf}(\dot{v_y} + v_x\dot{\psi} + \sin(\theta_y)) \tag{3.35}$$

$$\Delta F_{z,yr} = \mu_{z,yr}(\dot{v_y} + v_x\dot{\psi} + \sin(\theta_y)) \tag{3.36}$$

Here, $\mu_{z,yf}$ and $\mu_{z,yr}$ are called lateral load transfer coefficients for front and rear axles respectively and θ_y is road grade in lateral direction. More details on estimating these load transfer co-coefficients and plant model used for simulations are provided in [25]. For brevity, a brief discussion on how the load transfer coefficients are estimated and importance of these parameters in estimating vertical load on each tire is given below. The longitudinal load transfer coefficient is estimated by two sets of simulation results, in which first set of simulations are made while vehicle is operating at different levels of constant throttle openings with zero steering and braking commands. whereas for the second set of simulations, the vehicle is operated at different levels of constant brake commands with zero steering and throttle commands. i.e., the first set of simulations provide the effect of vehicle acceleration and second set of simulations provide effect of braking/deceleration on longitudinal load transfer.
Similarly for lateral load transfer coefficients, the simulations are made with sinusoidal variation of steering angle while keeping the longitudinal speed nearly constant or varying with very low frequency relative to steering angle variation frequency. These simulations essentially provide the effect of cornering on lateral load transfer. The equations from (3.31)-(3.36) and rearranging the v_x,v_y terms in non-linear tire model, the final dynamical safety constraint can be abbreviated as follows,

$$\mathcal{S}(v_x, v_y, \psi, \delta_f, a_x, \theta) \leq 0 \tag{3.37}$$

3.2.3.4 Equation (3.6):Stationary obstacle avoidance constraint

The obstacle avoidance constraint can be applied through simple distance formula, where the minimum distance between all the predicted vehicle positions and obstacles in the LIDAR view region should be more than a threshold value. This threshold value should be more than zero or any positive number. This creates a for-loop with k-iterations and k being the number of obstacles found in LIDAR region. In the present work, for the purpose of improving code performance, the maximum number of obstacles that can be detected in the LIDAR view region is restricted to

76. The following Pseudo code is developed for avoiding obstacles Therefore, the

Algorithm 1 Stationary Obstacle avoidance constraint

```
for $i = 1 : no\ of\ vehicle\ predictions(N)$ do
    for $j = 1 : no\ of\ obstacles(k)$ do
        $O_d = \sqrt{(P_i - O_m^{(j)})^2} \leq 7m$
    end for
end for
```

above obstacle avoidance can be abbreviated as follows,

$$\mathcal{O}_{st}[\mathcal{Z}(t), O_d^{(j)}(t)] \leq 0 \tag{3.38}$$

3.2.3.5 Equation (3.7):Prediction horizon constraint

It is critical to make prediction horizon with sufficient length to avoid obstacles that are not known priory in an unstructured environment. Therefore, to avoid any potential obstacles in advance, the vehicle states needs to be predicted for the minimum specified LIDAR view range L_d, given in Table. 3.2. This constraint is obtained through following equations,

$$L_p = v_x \times N \times dt \tag{3.39}$$

$$L_d - L_p \leq 0 \tag{3.40}$$

Here, L_p, v_x,N,dt represents the prediction horizon length in m ,vehicle longitudinal speed in m/s, number of predictions and time duration for each prediction respectively. Further, the prediction horizon window is defined using $T_p = N \times dt$ in *sec*. In CasADi problem formulation, it is difficult to vary the number of predictions N value than duration for each prediction dt. Therefore, the dt is varied based on the vehicle speed and keeps prediction horizon length at minimum specified LIDAR view range value. This constraint can be abbreviated in the following manner,

$$L[\mathcal{Z}, L_d, dt] \leq 0 \tag{3.41}$$

3.2.3.6 Equation (3.8):Stop constraint

This constraint simply ensures that the vehicle reached within the considerable limits of target location and vehicle can come to complete stop and reduces its predictions according to the target location. This constraint is only applicable, when the target position is within the LIDAR view range and both the soft and hard constraints be replaced with the below equations from

$$x_g - \epsilon \leq x(t) \leq x_g + \epsilon \tag{3.42}$$

$$y_g - \epsilon \leq y(t) \leq y_g + \epsilon \tag{3.43}$$

Here, ϵ is a small margin in m and vehicle said to be reached target, when it falls under this margin.The above constraint can be abbreviated as follows,

$$\mathcal{E}[\mathcal{Z}(t), \mathcal{Z}_{target}] \leq 0 \tag{3.44}$$

3.2.3.7 Cost function

The cost function is formulated for estimating the optimal steering angle and speed that would meet following requirements,
1. avoid obstacles and steep regions without priory information of them
2. reach the target as soon as possible
3. meet the vehicle safety requirements
4. Apply minimal control efforts.

As mentioned, the Non-linear MPC makes future predictions based on current vehicle position and thus the vehicle states are compared with target states during the prediction and at the end of prediction. The terms that are associated during the prediction called integral terms, whereas terms associated at the end of prediction called terminal cost terms. In the present work, there are total of eight terms including four integral cost terms and four terminal cost terms along with appropriate weights is given below,

$$\begin{aligned} J =& w_d \frac{d_f}{d_0} + w_\psi (\Delta\psi)^2 + w_t(T_p) + \int_0^k [\frac{w_{obs}}{O_{dp}}] \\ &+ w_{\psi_f} \int_0^{T_p} [\sin(\psi_g)(x - x_g) - \cos(\psi_g)(y - y_g)]^2 dt \\ &+ w_{f_z} \int_0^{T_p} [\tanh(-\frac{F_{zr-left} - a}{b}) \\ &+ \tanh(-\frac{F_{zr-right} - a}{b}) \\ &+ \tanh(-\frac{F_{zf-left} - a}{b}) + \tanh(-\frac{F_{zf-right} - a}{b})]dt \\ &+ \int_0^{T_p} w_Q [z(1:4) - z_{tar}(1:4)]dt \\ &+ w_c \int_0^{T_p} [w_\delta \delta_f^2 + w_J J_f^2 + w_\gamma \gamma_f^2]dt \end{aligned} \tag{3.45}$$

For brevity, the basic definitions of cost terms and their significance have been provided in the present work and more details can be found in [25]. Here, d_0 is the distance between vehicle current position $[x_0, y_0]$ to target position $[x_g, y_g]$ and d_f is the distance between vehicle final prediction position $[x(T_p), y(T_p)]$ to target position. The second terminal term $\Delta\psi$ is the difference between final prediction heading angle $\psi(T_p)$ and the angle of relative target heading, which is calculated from $[x(T_p), y(T_p)]$ relative to $[x_g, y_g]$ is given in below equation (3.46).

$$\begin{aligned} \Delta\psi &= atan2[\sin(\psi(T_p) - \psi_{rtg}), \cos(\psi(T_p) - \psi_{rtg})] \\ \psi_{rtg} &= atan2[(y_g - y(T_p)), (x_g - x(T_p))] \end{aligned} \tag{3.46}$$

The geometric representation of first three terminal terms in cost function indicates the final predicted vehicle position and angle should tend towards target with minimum time to reach the final predicted position. Further, the final terminal term provide soft constraint on stationary obstacle avoidance. Here, the O_{dp} indicates the distance between stationary obstacle and $x(T_p), y(T_p)$, and k represents the total number of obstacles detected in LIDAR view region. Therefore, with the appropriate weights, the predicted trajectory would avoid obstacles and reach target location as soon as possible.

In addition, there are four integral terms in which first term minimises the lateral error in predicted trajectory and the second term penalises the cost function, when the vertical load on each of the four tires is approaching the minimum specified load. This prevent the vehicle from unnecessary operation of near dynamical safety constraint. Here, parameters a, b are defined as follows

$$\begin{aligned} a &= F_{z_{thr}} + 3F_{z_{off}} \\ b &= F_{z_{off}} \end{aligned} \tag{3.47}$$

The values for $a, b, F_{z_{off}}$ and $F_{z_{thr}}$ are provided in Table. 3.2, and more details can be found in [24]. The third integral term minimises the error between the target vehicle states including x, y, ψ, V_x and corresponding prediction vehicle states to ensure the vehicle is tending towards target location. Therefore, the weight term w_Q is a 4×4 diagonal matrix, in which each diagonal element penalises the corresponding sate parameter. The final integral term represents the minimal control effort required to navigate the vehicle and penalises the cost function using weighting parameters w_δ, w_J, w_γ are given in Table.3.2.

3.3 Methodology for moving obstacles

As mentioned, the ABD-JDN algorithm can be applied to moving obstacle avoidance process with slight modification in its LIDAR detection process. Therefore, similar to stationary obstacle methodology, this section is mainly divided into three parts, in which part-1 provides moving obstacle detection process, part-2 provides moving obstacle state identification and avoidance process and part-3 presents the OCP formulation for moving obstacles and all three parts are explained in the following sections in detail,

3.3.1 Part-1: Detection process

The detections for moving obstacles are made with an angle of pi/12 from the vehicle straight view. Because, the obstacles either moving or stationary in straight section can be avoided through stationary obstacles avoidance algorithm and it is explained in previous sections. Further, the detection logic for moving obstacles is slightly different from stationary obstacle search logic. However, the logic still uses box detect method to find obstacles in the corresponding LIDAR view regions. It is assumed that, the vehicle mainly consists of three LIDAR modules with LIDAR-1 module being used for providing obstacle information in straight section, LIDAR-2 module provides obstacle information in vehicle right side direction and LIDAR-3 module provide vehicle left side obstacles information respectively. i.e., LIDAR-1 module is used for stationary obstacles avoidance algorithm and LIDAR-2 and LIDAR-3 modules are added to the moving obstacle analysis. Further, the LIDAR-2 and LIDAR-3 modules are assumed to be mounted with pi/12 angle from the vehicle straight section and the obstacle detection logic for LIDAR-2 and LIDAR-3 modules are processed as follows,
The obstacle detection or search logic for moving obstacles uses similar box detect logic from the previous stationary obstacle detection. It creates six box regions for obstacle search in both left and right directions of vehicle to mimic the LIDAR-2 and LIDAR-3 module regions. i.e.,LIDAR-2 module region consists of six boxes, that are $-\pi/12$ angle deviation from the vehicle current heading or straight path boxes. Similarly, LIDAR-3 module region consists of 6 boxes that are $\pi/12$ deviation from straight path as shown in Fig. 3.5. Therefore, the left and right LIDAR view regions are made with respect to the vehicle current heading and adding $\pi/12$ rad of angle in each side respectively. The moving obstacle detection logic first searches in right 6 boxes using slope search logic and then moves to left 6 boxes is shown in Fig. 3.5. When the obstacle is identified in either of box regions, the logic stores its position information in the temporary memory. The stored detected obstacle information is

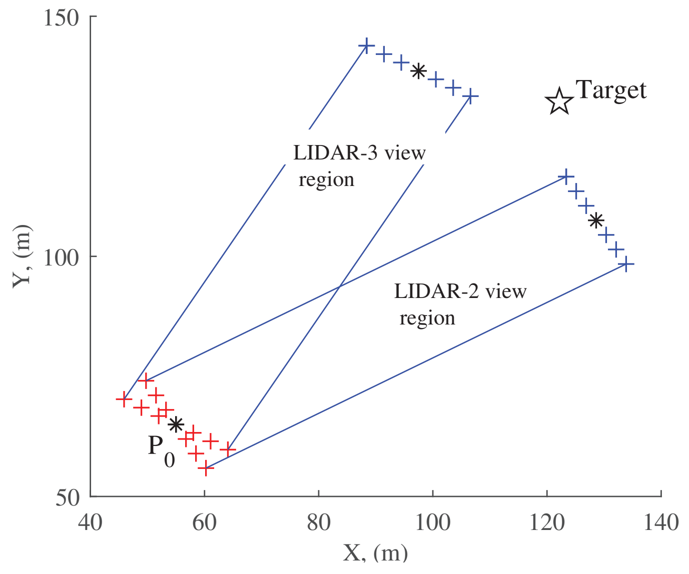


Figure 3.5: LIDAR view region for moving obstacle analysis

further processed to state identification section to know the obstacle speed and angle of shoot. The following sections provide detailed explanation on moving obstacle identification process and their state handling process in OCP problem formulation.

3.3.2 Part-2: State identification process

As the name suggests, in this section the logic estimates the obstacle states including position, speed and direction by using the vehicle state model and sensor model is explained below sections. The logic works based on the sensor and obstacle temporal information and the predictions of vehicle states. Here, it is assumed that, the LIDAR update rate is n times faster than MPC iteration or sampling time. During this sampling time period the logic makes certain calculations to confirm the obstacle movement and make its speed and heading estimations.

Vehicle state model

As mentioned, this model receives the obstacle position information from the stored detection process section. The vehicle state model provide vehicle prediction states based on the system dynamic model defined in equation 3.9. However, the predictions are made with the update rate of $dt_{mv} = \frac{T_s}{n}, sec$ and thus n prediction states are made in one MPC sampling period of $T_s, sec.$ Further, a distance calculation be made between the n prediction states $(P_{mv,i})$ and detected obstacle current/initial states $O^{(j)}_{m,0}$ and is given in eq. 3.48.

$$D_{V_{mobs}}[i,j] = \sqrt{(O^{(j)}_{m,0} - P_{mv,i})^2} \tag{3.48}$$

Here, i indicates number of predictions in one MPC iteration or sampling period T_s, where $(i = 1, 2, 3..n)$, and j is number of obstacles detected in moving obstacle LIDAR view region, where $(j = 1, 2...k)$, and each distance column array in $D_{V_{obs}}$ represents distance from individual detected obstacle current position to predicted vehicle states is shown in Fig. 3.6(a). Therefore, if there are k number of obstacles are detected with n vehicle prediction states, the distance matrix would become $[n \times k]$ matrix is given in the following matrix,

$$D_{V_{mobs}}[n,k] =$$

$$\begin{bmatrix} \sqrt{(O^{(1)}_{m,0} - P_{mv,0})^2} & . & . & \sqrt{(O^{(k)}_{m,0} - P_{mv,0})^2} \\ \sqrt{(O^{(1)}_{m,0} - P_{mv,1})^2} & . & . & \sqrt{(O^{(k)}_{m,0} - P_{mv,1})^2} \\ . & . & . & . \\ . & . & . & . \\ \sqrt{(O^{(1)}_{m,0} - P_{mv,n})^2} & . & . & \sqrt{(O^{(k)}_{m,0} - P_{mv,n})^2} \end{bmatrix} \tag{3.49}$$

As mentioned, In the present moving obstacle analysis, the number of predictions (n) and sensor measurements considered in one MPC sampling time period T_s is same and it is assumed as 5. Further, the maximum number of obstacles (k) can be detected in LIDAR view region is restricted to 75. The above Vehicle to obstacle distance array matrix $D_{V_{obs}}[n,k]$ is stored in a temporary memory for further comparison with sensor model and provide important information with respect to the moving obstacle identification and it is explained in following sensor model section.

Sensor model

As mentioned, the LIDAR sensor is assumed to be n times faster in providing obstacle information and thus make n measurements in one MPC iteration or sampling time period T_s. It is assumed that, the each measurement is made with equal interval and that coincide with the vehicle state model predictions $(P_{mv,i})$. Therefore, at each sensor measurement, it is assumed that the vehicle position changes and coincide with the vehicle model prediction states $(P_{mv,i})$. The time period dt_{mv} for vehicle predictions and LIDAR sensor measurements in moving obstacle analysis is too small and it is appropriate to assume the sensor measurements made at predicted vehicle states $(P_{mv,i})$. Similar to the vehicle state model, a distance calculation be made

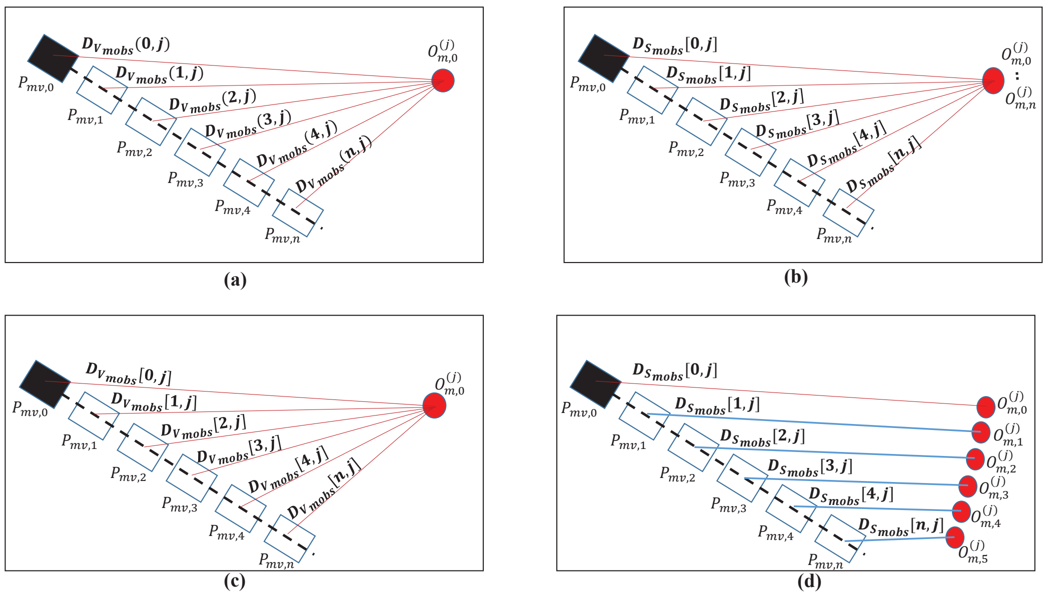


Figure 3.6: An illustration of moving obstacle state identification process in LIDAR view for moving obstacle zones; subplots (a) and (b) presents typical stationary obstacle detection process; subplots (c) and (d) presents typical moving obstacle detection process

between vehicle prediction states and measurement states is given in below (3.50).

$$D_{S_{mobs}}[i,j] = \sqrt{(O_{m,i}^{(j)} - P_{mv,i})^2} \tag{3.50}$$

Here, each column in $D_{S_{mobs}}$ represents measured obstacle distance from vehicle prediction states and can be written in matrix form as follows,

$$D_{S_{mobs}}[n,k] =$$

$$\begin{bmatrix} \sqrt{(O_{m,0}^{(1)} - P_{mv,0})^2} & . & . & \sqrt{(O_{m,0}^{(k)} - P_{mv,0})^2} \\ \sqrt{(O_{m,1}^{(1)} - P_{mv,1})^2} & . & . & \sqrt{(O_{m,1}^{(k)} - P_{mv,1})^2} \\ . & . & . & . \\ . & . & . & . \\ \sqrt{(O_{m,n}^{(1)} - P_{mv,n})^2} & . & . & \sqrt{(O_{m,n}^{(k)} - P_{mv,n})^2} \end{bmatrix} \tag{3.51}$$

Further, each row in $D_{S_{mobs}}$ represents the sensor measurement updated distance with the corresponding vehicle predicted position $P_{mv,i}$. Therefore, when the obstacle is stationary the final row of sensor model distance matrix matches with final row of vehicle state model. Therefore, the condition for moving obstacle is defined as follows, (3.50).

$$D_{V_{mobs}}[n^{th}, k^{th}] - D_{S_{mobs}}[n^{th}, j^{th}] > St_{thr} \tag{3.52}$$

Here, St_{thr} is a small positive number in distance that would compensate for the

LIDAR sensor measurement noise. Once, it is confirmed that the obstacle is moving, the further analysis for finding corresponding obstacle speed and angle of shoot is made through following moving obstacle state handling section.

3.3.2.1 Moving Obstacle state handling process

The below equations from (3.53)- (3.55) provide moving obstacle state information including speed, heading and position.

$$[x_{mobs}^{(j)}, y_{mobs}^{(j)}] = O_{m,n}^{(j)} \tag{3.53}$$

$$v_{mobs}^{(j)} = \left(\sqrt{(O_{m,n}^{(j)} - P_{mv,n})^2} - \sqrt{(O_{m,0}^{(j)} - P_{mv,0})^2}\right)/dt_{mv} \tag{3.54}$$

$$\psi_{mobs}^{(j)} = atan2(O_{m,n}^{(j)}, O_{m,0}^{(j)}) \tag{3.55}$$

i.e, During one sampling time period t, the difference between initial and final measurements of LIDAR sensor provide corresponding obstacle speed and heading direction estimates. Based on these obstacle state estimations and LIDAR detection process, the state handling process determines the detected obstacle is in dangerous zone through the equations (3.56) and (3.57).

$$\psi_{mobs}^{(j)} > \psi \ \ (or) \ \ \psi_{mobs}^{(j)} < \psi_{ulimit} \tag{3.56}$$

$$\psi_{mobs}^{(j)} < \psi \ \ (or) \ \ \psi_{mobs}^{(j)} > \psi_{llimit} \tag{3.57}$$

Here,

$$\psi_{ulimit} = (\psi + (\pi + \pi/9))$$
$$\psi_{llimit} = (\psi - (\pi + \pi/9))$$

Once the algorithm determines that the detected moving obstacle is in dangerous zone, the corresponding estimates are stored in parameters that would later be used for updating moving obstacle states in NMPC problem formulation. The predictions of obstacles and the corresponding constraints are explained in NMPC with moving obstacle problem formulation. Further, the obstacles with road grade avoidance process is still applicable and it is detailed in section.3.2.2.5.

3.3.3 Part-3: Optimal control Problem formulation for moving obstacles

The optimal control problem formulation for moving obstacles is an extension to the existing stationary obstacle formulation. The cost function and constraints are still valid for the moving obstacle formulation too. However, there are two major aspects have been incorporated to the stationary obstacle formulation and it is explained in the following session,

Vehicle dynamic constraint for moving obstacle

The first major change in moving obstacle problem formulation is, incorporating moving obstacle model into vehicle model using obstacle speed and angle of shoot information is given in A_{mv} matrix. i.e., in-addition to vehicle states the moving obstacle states are predicted for the given horizon length. As mentioned, the obstacle speed and angle of shoot are obtained through Moving obstacle LIDAR process at each MPC iteration and this information is fused with vehicle state model to make predictions.
Therefore, the vehicle dynamic state model for moving obstacle is defined as, Where,

$$A(z) = \begin{bmatrix} v_x \cos\psi - (v_y + l_f\dot{\psi})\sin\psi \\ v_x \sin\psi + (v_y + l_f\dot{\psi})\cos\psi \\ \dot{\psi} + (v_x/L)\tan\delta_f \\ a_x \\ (F_{yf} + F_{yr})/M_{total} - v_x\dot{\psi} \\ (F_{yf}l_f - F_{yr}l_r)/I_z \\ v_{mobs}^{(1)} \cos\psi_{mobs}^{(1)} \\ v_{mobs}^{(1)} \sin\psi_{mobs}^{(1)} \\ v_{mobs}^{(2)} \cos\psi_{mobs}^{(2)} \\ v_{mobs}^{(2)} \sin\psi_{mobs}^{(2)} \\ v_{mobs}^{(3)} \cos\psi_{mobs}^{(3)} \\ v_{mobs}^{(3)} \sin\psi_{mobs}^{(3)} \\ 0 \\ 0 \end{bmatrix}$$

$$B = \begin{pmatrix} 0_{2\times 6} & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \end{pmatrix}$$

Here, v^j_{mobs},ψ^j_{mobs} are moving obstacle longitudinal speed and angle of shoot respectively and here, $j = 1, 2, ..k$, represents the number of moving obstacles detected in the region. The algorithm can easily be expanded to any number of moving obstacles. However, due to computational limits, the present study can handle maximum of three moving obstacles in the given vehicle view region. Further, from constraint equation (3.7), the prediction horizon depends on vehicle longitudinal speed than moving obstacle speed and thus the prediction length for moving obstacles is not fixed and it would vary based on the vehicle speed v_x.

Moving obstacle avoidance constraint

Once, the moving obstacle state predictions are made, it is critical to develop a constraint that would avoid these predictions from vehicle collision. Therefore,the second major change in moving obstacle formulation is adding the extra moving obstacle constraint to the existing algorithm and it is explained with the below Pseudo 2. Therefore, the above obstacle avoidance can be abbreviated as follows,

Algorithm 2 Moving Obstacle avoidance constraint

```
for $i = 1 : no\ of\ vehicle\ predictions$ do
  for $j = 1 : no\ of\ moving\ obstacles$ do
    for $i_m = 1 : no\ of\ moving\ obstacle\ predictions$ do
      $O_{d_{mobs}} = \sqrt{P(i) - O^{(j)}_{(p,i_m)})^2} \leq 7m$
    end for
  end for
end for
```

$$\mathcal{O}_{mobs}[\mathcal{Z}(t), O^{(j)}_{mobs}(t)] \leq 0 \tag{3.58}$$

From the above two sections, the problem formulation for moving obstacles can be The NMPC problem formulation along with moving obstacles constraints are formulated using casadi software [55] which is explained in the following ‘N-MPC problem formulation using Casadi tool’ section.

3.4 N-MPC problem formulation using Casadi tool

The present work uses open source numerical optimization tool called CasADi, for formulating the optimal control problem [55]. The CasADi tool is efficient in solving Non-linear programming problems and best suits for the present N-MPC obstacle avoidance application. The CasADi uses symbolic framework to formulate cost function, constraints and use these to define automatically differentiable functions. Further, for better convergence direct multiple shooting method is implemented in the present CasADi formulation framework. However, the CasAdi is not a mathematical solver tool, instead it set up the problem to feed it into external solver for the better and fast results. The user can chose any external solver based on the intricacy of the problem. However, in the present work Interior point method (IPOPT) [56] solver is used for solving the obstacle avoidance optimization. The following steps are used for formulating the point stabilization problem in CasADi via full featured MATLAB coding

1). Define all design variables including states and control inputs using CasADi symbolic framework.

2). Formulate Objective function (J) using above defined symbolic design variables.

3). Define constraints using design variables, g. This includes ODE formulation using CasADi function definition. Here, some of the constraints be updated at each iteration based on the updated optimization variables. For instance, in the current problem the constraints on acceleration would be updated at each iteration based on current vehicle speed design variable.

4). Define optimization or manipulated variables, w. These variables would be updated at each iteration and at each node (for multiple shooting method) to reach the target point.

5). Define initial and target variables P for the point stabilization problem. In the present work, the target values may vary temporarily during the navigation for avoiding the obstacles. However, the final objective target location would not be changed. For instance, the target heading and velocity can be updated at each iteration based on detected obstacle distance and road grade indices. Therefore, vehicle can slow down and take aversion for avoiding these obstacles. However, the final target point at which vehicle need to arrive would not be changed.

6). Formulate the Non-linear MPC problem using J, g, w, P variables in struct and chose the solver. As mentioned, in the present work, the IPOPT solver is used for solving the optimization problem [56].

7). Define solver settings including convergence criteria,maximum iterations,acceptable objective change tolerance and print level.

8). Assign the above problem formulation and solver to an object S using *nlpsol* syntax. The defined object S is used for accepting the updated design and target variables at each iteration and provides the least cost optimization control commands for navigating vehicle through obstacle-grade fields.

Unlike the conventional methods, from steps 1 to 7, all terms including cost function, design variables, constraints are defined symbolically and therefore the optimization problem would be solved in efficient manner by saving physical memory allocation on each variable. This way, CasADi tool provide convenient way to define target variables, constraints, cost terms and assign custom solver for solving complex optimization problems.

3.5 Results and Discussion

The results and discussion section is mainly divided into 2 sections, in which first section provide results on avoiding stationary obstacles along with steep regions and later section incorporates moving obstacle avoidance to the existing results.

3.5.1 Results on stationary obstacles and steep region (bumps) avoidance

This section deals with simulation results of developed non-linear MPC stationary obstacle avoidance algorithm with road grade included the model. In this section, six simulations with five paths have been considered to observe the performance of the algorithm from simple through relatively complex obstacle field scenarios. Out of five scenarios, first three simulations have made for constant speed and variable speed conditions with no road grade (flat road condition) in the model. The constant speed used in the present work is 20 m/s and in the variable speed condition, it can be varied from 5 to 29 m/sec. These simulations are made to compare the effect of

Table 3.2
N-MPC optimization parameters

Parameter	Value	Units
M_{total}	2869	kg
I_z	4110	$kg - m^2$
L_d	75	m
l_f, l_r	1.58,1.72	kg
$\mu_{z,x}$	806	$N/(m/s^2)$
$\mu_{z,yf}$	675	$N/(m/s^2)$
$\mu_{z,yr}$	1076	$N/(m/s^2)$
$F_{z,thr}$	995	N
a, b	1300,100	N
$[\delta_{f,min}, \delta_{f,max}]$	[-30,30]	$^\circ$
$[v_{x,min}, v_{x,max}]$	[5,29]	m/s
$[J_{f,min}, J_{f,max}]$	[-5,5]	m/s^3
$[\gamma_{f,min}, \gamma_{f,max}]$	[-5,5]	$^\circ/s$
$[O_{d,min}, O_{d,max}]$	[7,∞]	m
St_{thr}	5	cm
$\theta_{critical}$	15	$^\circ$
$[\theta_{thr,min}, \theta_{thr,max}]$	[6.8,13.8]	$^\circ$
$w_d, w_{psi}, w_{obs}, w_t$	0.5,1e+1,1e-3,5e-3	$-$
w_{psi_f}, w_{f_z}, w_c	5e-2,5e-3,1	$-$
$w_Q[1,1], w_Q[2,2]$	5e-2,5e-2	$-$
$w_Q[3,3], w_Q[4,4]$	6e+1,5e+1	$-$
w_δ, w_J, w_γ	5e-3,5e-2,5e-2	$-$

constant speed and variable speed conditions and thus to make sure both conditions reach target safely. The remaining two simulations are made to show the performance of algorithm that is capable of maneuvering the vehicle through steep/bumpy regions along with complex obstacle field. Here two types of obstacles are considered, one with constant 5 m diameter called individual obstacle and other with lengthy obstacle with different shapes. The complex field can be defined with density of individual obstacles and lengthy obstacles with varying sizes.
The road grade is made using contour plot with the maximum value of 14° and mimics the real road conditions is shown in Fig.3.7. The map can be replicated in negative grade direction and the results are still valid for the given range of grade magnitude. The grade between the contour lines are interpolated for the actual grade value and handling of road for the avoidance is explained in ABD-JDN algorithm. As mentioned, the road grade below 6.8 ° is considered as flat road condition and in the first 3 set of simulation results, the contour plot for grade is not included. i.e.,the white space background in the first 3 simulation results indicates the results with flat

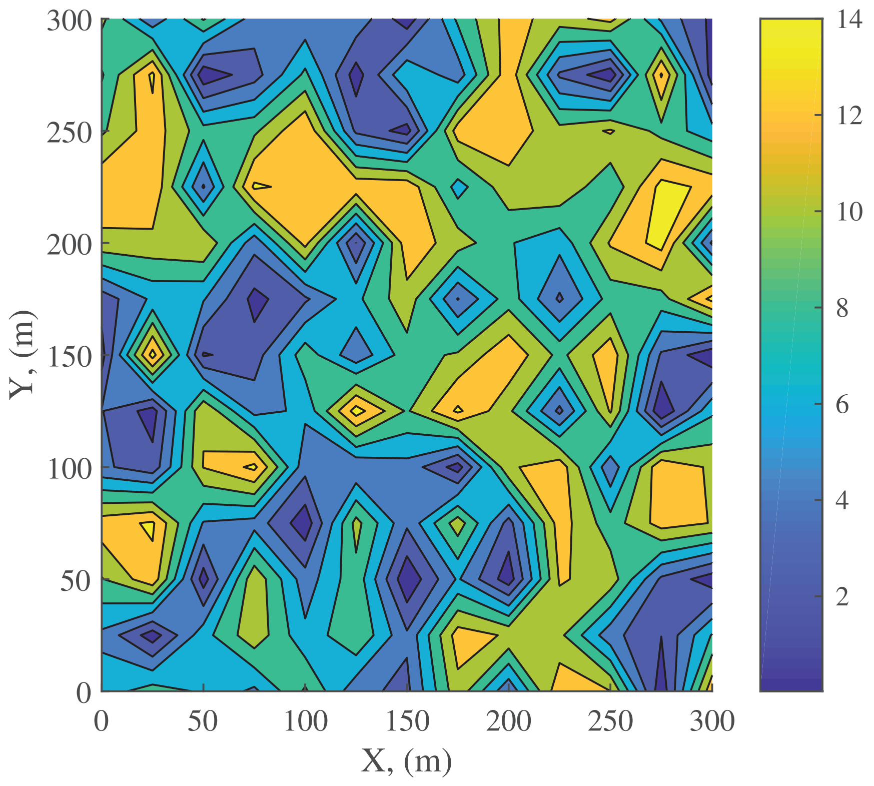


Figure 3.7: A typical steep region construction in the off-road navigation

road condition. Table. 3.2. shows the details of lower, upper and critical threshold values used in the study and as mentioned, the vehicle treats any road grade value that is above critical value as obstacle. However, the vehicle can still pass through the regions that are lower than critical grade value to optimize the target path.
In the simulation, the initial speed of vehicle is fixed at 20 m/sec and LIDAR detection range is assumed to be 75 m from the vehicle front position. The final objective of algorithm is to avoid obstacles and steep regions that are above critical value with a minimum distance of 8 m from the vehicle current position by using the obstacle search algorithm. However, the algorithm tries to avoid steep regions that are below critical value by meeting the vehicle safety and optimum path constraints. Though the LIDAR modules can process the 90° vehicle view, the obstacle search algorithm restricts its fusion processing, when it is unnecessary. i.e, the algorithm divides the vehicle view into sections and if vehicle can find a way through straight section, the algorithm omits the sensor fusion analysis from other modules. Further, the obstacle search logic stores all the detected obstacles information that are within the range of 125 m from the current vehicle position. Therefore, the MPC algorithm consider this information to formulate control commands and thus it is unlikely to fail even at constant speed complex obstacle scenarios [25]. However, this might cause vehicle to take longer paths and slower speeds for extended period.
At every iteration, the MPC generates optimal trajectory into the future that would

avoid both obstacles and bumps in the vehicle view region. From this optimal control trajectory, only the first sample of control commands are executed for the vehicle navigation and remaining control trajectory would be used for initialization of prediction matrix for next iteration. A new trajectory would be updated in next iteration, which consider the vehicle current updated states and obstacles information from sensors. The algorithm considers 0.3 sec as update period and further executes the control commands for 0.3 sec as well. However, the control commands may not be constant during this execution time period and the sampling time for control command is 0.05 sec.

3.5.1.1 Simulation-1 results

As mentioned, the first three simulations in this section are made for assessing the algorithm performance for both constant speed and variable speed cases as shown in Fig. 3.8 to Fig. 3.13. In the first two scenarios, both constant and variable speed conditions reached the target position by meeting all vehicle safety constraints. In this, the first scenario is relatively simple in terms of obstacle field complexity is shown in Fig. 3.8. From the Fig. 3.8(c) it can be clearly observed that the variable speed condition gradually increases the vehicle speed to maximum limit when vehicle is free from obstacle field and reaches target location 3.5 sec earlier than constant speed condition. This makes it about 7% faster compared to constant speed condition is shown in Fig. 3.8(b). Though the constant speed problem formulation is relatively simple, the variable speed condition exploits the full range powertrain capability and thus reaches the target location as soon as possible.

3.5.1.2 Simulation-2 results

In the second scenario, the vehicle passes through relatively moderate obstacle field and both constant speed and variable speed logic reaches target location safely with slightly different paths is shown in Fig. 3.10. In this scenario, though the constant speed condition reaches faster than variable speed condition, the path taken by variable speed is optimal path. Further, variable speed condition uses obstacle search logic to vary the vehicle speed when it detects obstacles in the vehicle view region and it assign more weightage towards safety than reaching the target location faster. However as mentioned, when the vehicle is not encountered with obstacles the vehicle starts accelerating and reaches to its maximum speed and in the long run variable speed condition could reach faster than constant speed condition. More importantly,

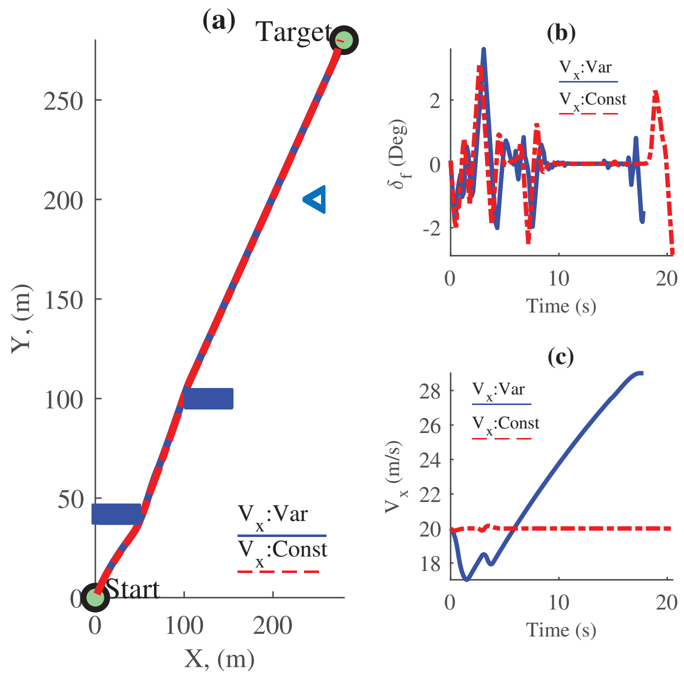


Figure 3.8: simulation-1 results, with relatively simple obstacle field: (a) Path followed by AGV for both variable and constant speed scenarios;(b) Steering commands generated by MPC for variable and constant speed scenarios; (c) Speed profiles for variable and constant speed scenarios.

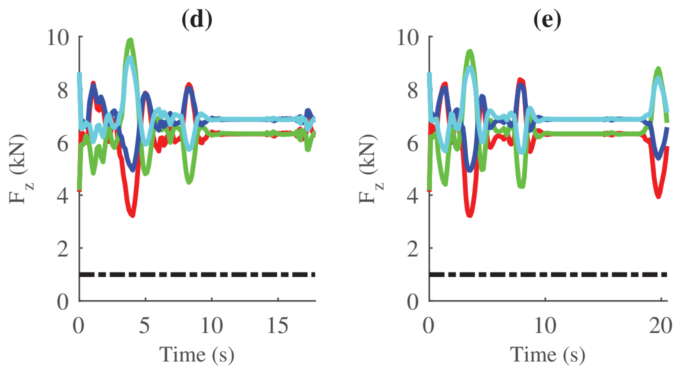


Figure 3.9: simulation-1 results: (d) Wheel lift-off constraint for variable speed scenario;(e) Wheel lift off constraint for constant speed scenario.

the systematic formulation of vehicle speed control not only avoids conservative operation of powertrain but also passes through complex obstacle field safely without collision. However, the constant speed approach fails to meet vehicle safety requirements when it passes through complex field and it is explained in scenario 3.

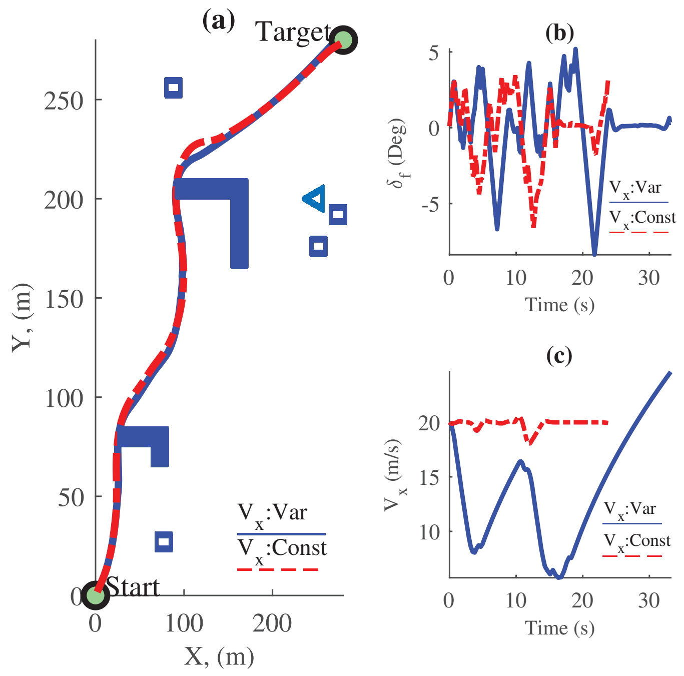


Figure 3.10: simulation-2 results, with relatively moderate obstacle field: (a) Path followed by AGV for both variable and constant speed scenarios;(b) Steering commands generated by MPC for variable and constant speed scenarios; (c) Speed profiles for variable and constant speed scenarios.

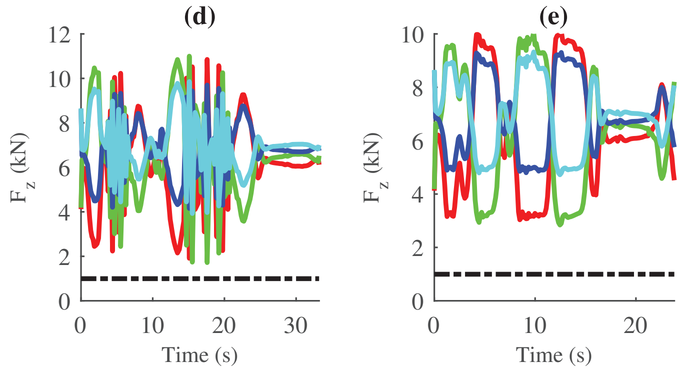


Figure 3.11: simulation-2 results: (d) Wheel lift-off constraint for variable speed scenario;(e) Wheel lift off constraint for constant speed scenario.

3.5.1.3 Simulation-3 results

The third set of simulations are made for the vehicle to pass through complex obstacle field, in which vehicle is trapped into enclosed surroundings and creates a necessary condition to reduce the vehicle speed to avoid obstacle field is shown in Fig. 3.12.

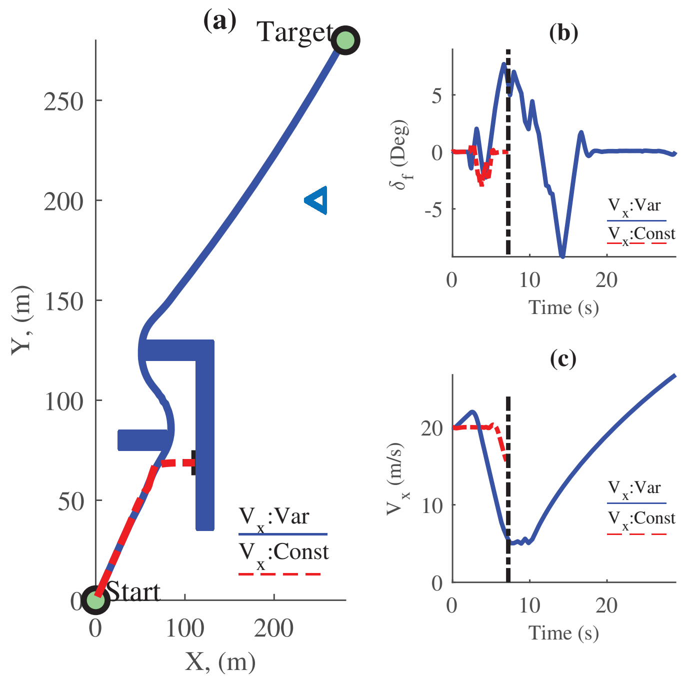


Figure 3.12: simulation-3 results, with relatively complex obstacle field: (a) Path followed by AGV for both variable and constant speed scenarios; however, the constant speed scenario fails to reach target location safely. (b) Steering commands generated by MPC for variable and constant speed scenarios; (c) Speed profiles for variable and constant speed scenarios.

From the simulation results, it is confirmed that the constant speed condition could not reach target location and crashes into obstacle field at 7.9 sec is shown in Fig. 3.12(a) to (c). on the otherhand, the variable speed condition approaches obstacle field with caution by reducing the vehicle speed according to the obstacle distance and accelerates as soon as vehicle finds obstacle free region. Here, vehicle speed during the trapped region reaches to about 6 m/s and maneuvers the vehicle in obstacle-free region in controlled manner. From Fig. 3.13, it can be observed that the vertical load on all four tires is more than specified threshold value and thus vehicle dynamical safety is ensured for the whole trip. Here black dotted line indicates the specified minimum threshold load for the vehicle dynamical safety.

3.5.1.4 Simulation-4 results

These simulations incorporates road grade information to the existing obstacle field and assess the performance of the obstacle-grade avoidance algorithm for

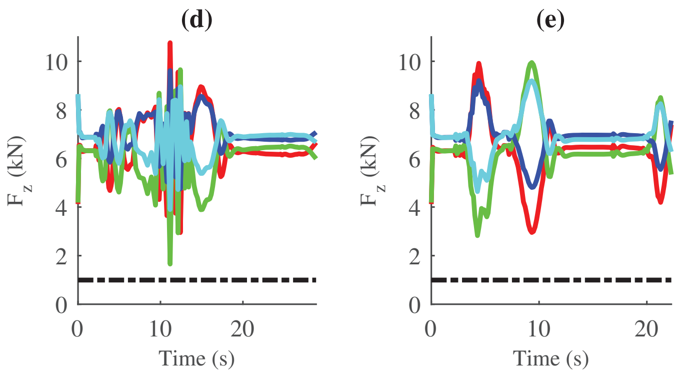


Figure 3.13: simulation-3 results: (d) Wheel lift-off constraint for variable speed scenario;(e) Wheel lift off constraint for constant speed scenario.

off-road applications. Three scenarios have been considered to study the algorithm performance and in which first set of simulations includes moderate obstacle field along with different road grade conditions. Second set of simulations include relatively complex obstacle field along with road grades and final set of simulations are made for demonstrating the algorithm robustness to the vehicle state estimation uncertainty with the existing second scenario obstacle field is shown in Fig. 3.14 to Fig. 3.19 Further, all of these obstacles and road grades are included in such a manner that they would obstruct the vehicle optimal path and algorithm has to reach target location by meeting all vehicle safety constraints with collision free and yet reach the target as soon as possible. i.e., vehicle has to avoid obstacles but it is acceptable to pass through grade regions to reach the target with optimal path. However, during this optimal path, vehicle need to ensure vehicle dynamical safety constraint by meeting minimum threshold load on all four wheels. As mentioned, the road grade can make substantial longitudinal load transfer and thus it is important to consider all four wheels for meeting the vehicle dynamical safety.

As mentioned, the fourth set of simulations in this section are made for the vehicle to pass through relatively moderate obstacle field with grades along the path is shown in Fig. 3.14(a) From the simulation results, it is confirmed that, the vehicle avoids all the obstacles and tries to move away from bumps or steep regions on the road. As mentioned, the road-grades in the path mimic real-world grades and the regions between contour lines are interpolated. In the structured traffic lane road conditions, the road grade varies between 0 to 6.8 deg and therefore, the vehicle considers these grade conditions are safe regions and the vehicle tries to go through these blue regions. Further, the road grade that is more than critical value is considered as obstacle field and vehicle need to avoid these regions. These regions are colored with dark yellow. From Fig. 3.14(a), it can be observed that, the vehicle passes through blue puddles upto [150m,125m] zone and after this vehicle encounters with both obstacle and steep region. The vehicle need to satisfy obstacle avoidance as well as optimal path condition and thus vehicle tries to pass through away from

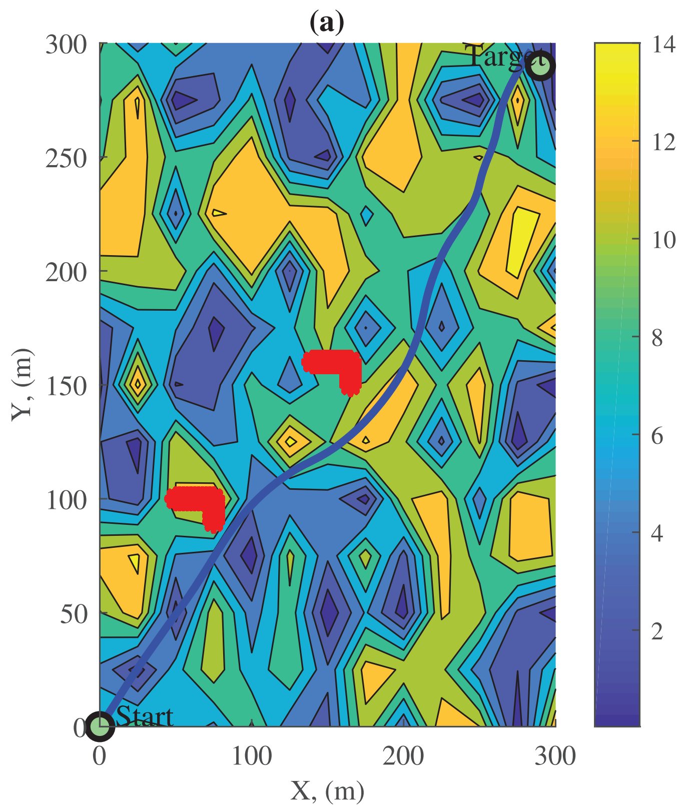


Figure 3.14: simulation-4 results, with relatively moderate obstacle-road grade field: (a) Path followed by AGV, which avoids both obstacles and road grade regions.

obstacle and yet reaches the target location as soon as possible. This condition shows that, vehicle can pass through steep regions to meet optimal path condition. However, vehicle has to ensure safety by meeting the minimum safety constraint load of $995N$ at all the times during the navigation is shown in Fig. 3.14(e). The corresponding vehicle speed, heading and steering wheel commands are provided in Fig. 3.14(b) to Fig. 3.14(d).

3.5.1.5 Simulation-5 results

The fifth set of simulations are made for running the vehicle through lengthy obstacles as well as randomly distributed 30 individual obstacles along with bumps/road grade on the road is shown in Fig. 3.16(a). From the result, it can be seen that the vehicle reaches target point by avoiding both obstacles and bumps that are on the way by meeting the vehicle safety constraints at each iteration. However, at [200m,150m]

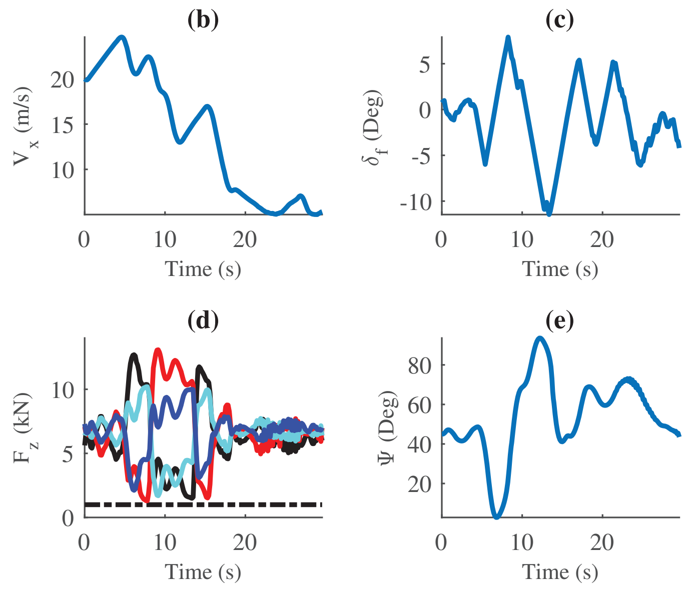


Figure 3.15: simulation-4 results: (b) and (c) Speed and Steering commands generated by MPC for obstacle-road grade avoidance; (d) Wheel lift-off constraint and vehicle heading variation for obstacle-road grade avoidance.

location, the vehicle seem passing through the steep region. This might be due to more weightage towards optimal path than steep region avoidance and vehicle does not find obstacle due to grade regions in the encountered grade puddle at this location. This makes vehicle to pass through risky regions and reach the target location as soon as possible. Therefore, the unnecessary acceleration and deceleration scenarios can be avoided during the navigation is shown if Fig.3.17(b). However, during the process vehicle has to ensure the dynamical safety and it is confirmed through Fig. 3.16(a) to 3.16(e). This scenario indicates that the vehicle can pass through moderate steep regions to avoid obstacles and meeting optimal path by ensuring the dynamical safety.

3.5.1.6 Simulation-6 results

As mentioned, the final set of simulations in this section is made for studying the inherent robustness of MPC problem formulation for avoiding obstacles and steep regions with vehicle position uncertainty. However, the uncertainty values are considered to mimic the real-world sensor measurements. In general, the typical GPS sensor with base station capability, would have the error of $\pm 0.3m$ in vehicle position estimation. Therefore, the Gaussian white noise with the magnitude of 0.3 m error

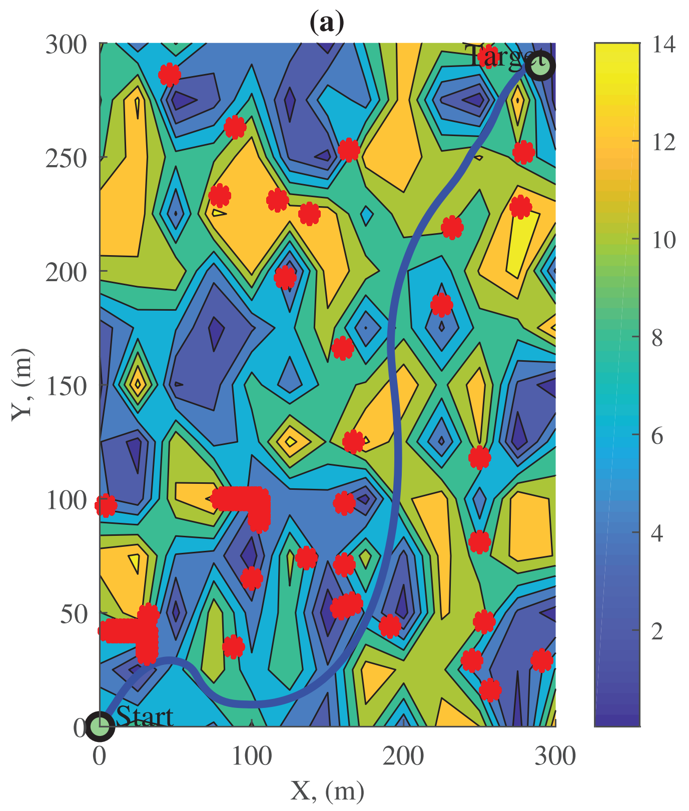


Figure 3.16: simulation-5 results, with relatively complex obstacle-road grade field: (a) Path followed by AGV, which avoids both individual and lengthy obstacles and road grade regions; however, the vehicle can still pass though moderate road grade regions to meet safety criteria and optimal path.

is introduced into the system. The typical inertial measurement sensor (IMU) unit provide measurements with the uncertainty of $\pm 2.5^\circ$ and same is applied to the vehicle heading uncertainty in terms of Gaussian white noise. Further, $\pm 0.1 m/s$ error in vehicle speed and $\pm 5cm$ error in obstacle distance is applied. The total of 30 simulations are made with the previous scenario-2 obstacle field given in Fig. 3.14(a). In all of the 30 runs, the vehicle avoids both obstacles and bumps but followed different routes to reach the target point. However, this is acceptable to meet the vehicle safety and obstacle minimum distance constraints and reach the target point as soon as possible. From Fig. 3.18(a), it is observed that, some of the simulations the vehicle reduces its speed to minimum of $5 m/s$ as it approaches the obstacle field region, and accelerates as soon as it finds obstacle free region to reach target point with optimal path. Further, in all of these conditions, the vehicle ensures the dynamical safety by meeting the minimum specified vertical load on all four tires is shown in Fig. 3.18(d). From the above 30 simulations, it is confirmed that the MPC formulation for obstacle-grade avoidance logic is robust enough for considered uncertainty limits. However, for more significant uncertainty the algorithm may not be robust and may

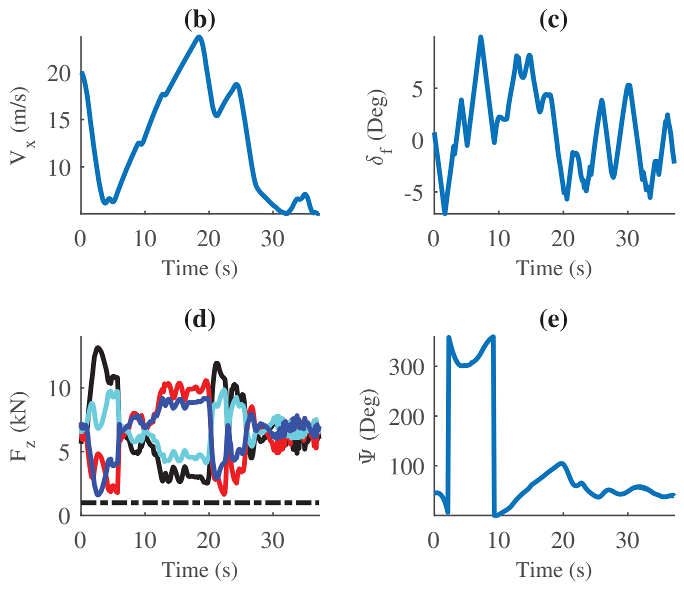


Figure 3.17: simulation-5 results: (b) and (c) Speed and Steering commands generated by MPC for complex obstacle-road grade field avoidance; (d) Wheel lift-off constraint and vehicle heading variation for complex obstacle-road grade field avoidance.

be required to develop a robustness scheme and it is left for further research work.

3.5.2 Results on moving obstacle avoidance

This section incorporates moving obstacles to the existing section.3.5.1. obstacle field, and it is mainly divided into two scenarios. The first scenario considers the stationary obstacle field that mimics the scenario-2 in section-1 and adds three moving obstacles at different locations with different angle of shoot. This scenario consider the road is flat and no contour plot for grade is included in the plots. Similarly, the second scenario consider the obstacle field, that is similar to scenario-2 in section-2 and it includes road grade, three moving obstacles at different locations with different angle of shoot. The location and angle of shoot for each moving obstacle is considered in such a way that the vehicle gets trapped into moving obstacles path and they would make collision, if vehicle follows the existing stationary obstacle path. However, in both of the above scenarios, the moving obstacle algorithm could able to deal with both stationary and moving obstacles and reached the target location without collision and it is detailed in the following sections.

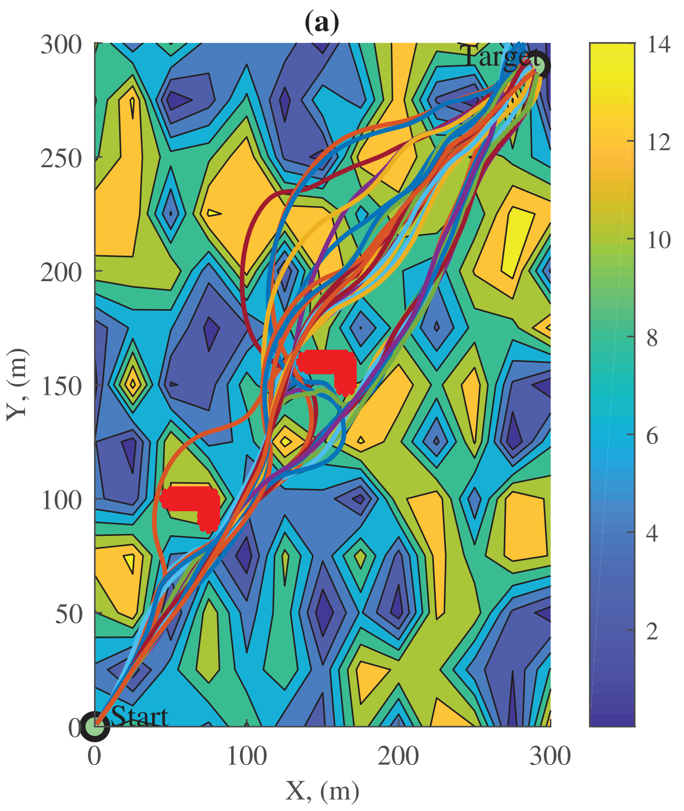


Figure 3.18: simulation-6 results for uncertainty analysis with relatively moderate obstacle-road grade field: (a) Path followed by AGV with thirty simulations, in which all simulations avoid both obstacles and road grade regions.

3.5.2.1 Simulation-1 results

As mentioned, the scenario-1 considers three moving obstacles with two lengthy stationary obstacles in the path is shown in Fig. 3.20(a). The Table. 3.3, provide the values for three moving obstacle initial states and their corresponding angle of shoot and speed. The black line with arrow in Fig. 3.20(a) indicates the direction and path followed by the moving obstacles during the simulation. From Fig. 3.10(a) and Fig. 3.20(a), it can be clearly observed that, for the similar stationary obstacle field the vehicle should have followed the section-1 scenario-2 route but made different route due to the interference of moving obstacles. The vehicle made major change in route/path at about $[110m, 110m]$ due to the first and second moving obstacles interference and it is detailed in Fig. 3.22. The Fig. 3.22, provide series of nine plots that captures the major changes in vehicle path due to moving obstacles. Each plot or frame in the figure defines one MPC iteration and the plots are referred in sequence from 1 to 9 from left upper corner to lower right corner respectively. In the

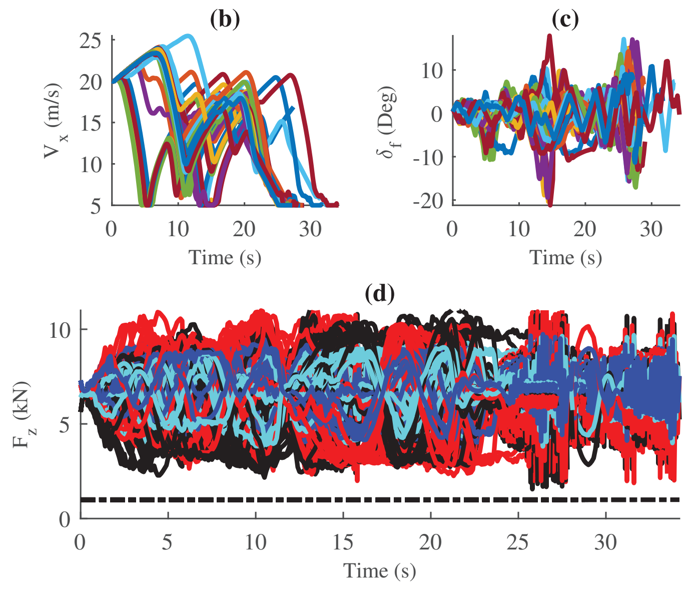


Figure 3.19: simulation-6 results: (b) and (c) Speed and Steering commands generated by MPC for thirty corresponding simulations in (a); (d) Wheel lift-off constraint for thirty corresponding simulations in (a), and all simulations meet the safety requirements.

Table 3.3
Moving obstacle initial state values for the simulation

Parameter	Value	Units
$[x^{(1)}_{mobs_{init}}, y^{(1)}_{mobs_{init}}]$	[200,150]	m
$v^{(1)}_{mobs}$	10	m/s
$\psi^{(1)}_{mobs}$	$(\pi + \pi/18)$	rad
$[x^{(2)}_{mobs_{init}}, y^{(2)}_{mobs_{init}}]$	[30,200]	m
$v^{(2)}_{mobs}$	7	m/s
$\psi^{(2)}_{mobs}$	$(2\pi - \pi/4)$	rad
$[x^{(3)}_{mobs_{init}}, y^{(3)}_{mobs_{init}}]$	[100,250]	m
$v^{(3)}_{mobs}$	10	m/s
$\psi^{(3)}_{mobs}$	$(3\pi/2{+}pi/36)$	rad

first five frames/plots, the vehicle detects the lengthy obstacle and tries to avoid with minimal distance. As mentioned earlier, the algorithm stores obstacle information, when it detects obstacles in the vehicle view region and deletes once the vehicle move 125 m away from it. Here, the detected obstacles are shown in red color and blue color dots indicates the undetected obstacles. Until frame 5, the vehicle follows same path as stationary obstacle path. However, at frame 6 algorithm detected the first moving obstacle and it estimated the moving obstacle speed and its future position

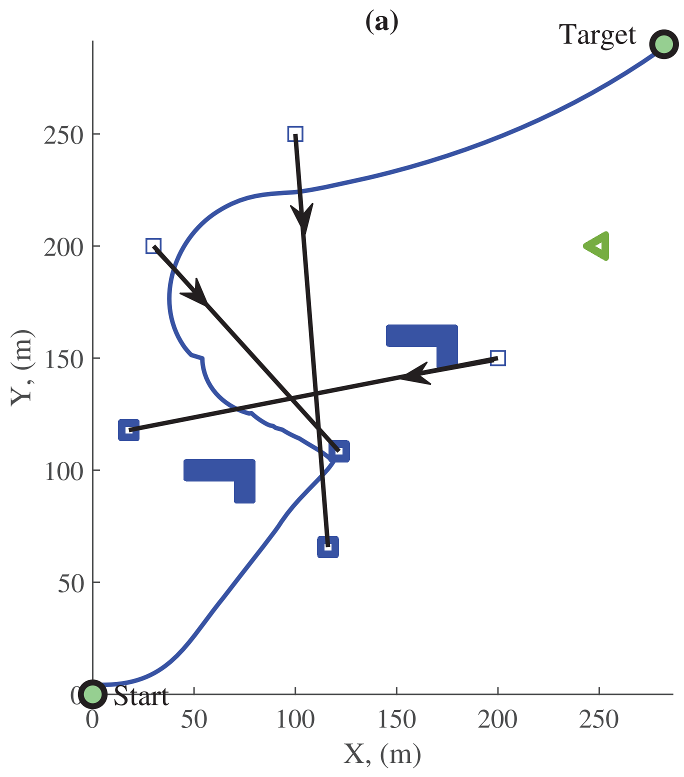


Figure 3.20: simulation-1 results for moving obstacle analysis with relatively moderate obstacle field: (a) Path followed by AGV, which avoids both moving and stationary obstacles.

predictions. At this point, the algorithm makes vehicle state predictions that must be avoided with any of the obstacle predictions. i.e., each vehicle state prediction has to maintain a minimum specified distance from all moving obstacle state predictions to avoid the collision. This minimum distance constraint averts the vehicle path towards left direction is shown in Fig. 3.22 (frame 6). At this point, the corresponding vehicle speed reduces to 14 m/s to ensure dynamical safety. Further, from frame 7 to 9, the algorithm detects second moving obstacle that is in vehicle dangerous zone. The dangerous zone for moving obstacles is defined in previous section.3.3. From frames 7 to 9, the second moving obstacle persists the vehicle to take further left turn for longer periods to avoid the future predictions collision. However, this further left path increases the target distance and vehicle would always tend to make right turn to reach target as soon as possible with optimal path. This behavior can be explained in the following session using Fig. 3.23. The Fig. 3.23 has made with series of nine events that started from 125*sec*, in which another major path change decision has been made. As mentioned, the vehicle would always tend to follow optimal path and tries to get back as soon as it finds obstacle free region. Therefore, from frame-1 to frame-3 in Fig. 3.23, the MPC algorithm commands vehicle to take right turn while it was navigating towards left direction. However, in frame-4 the algorithm again

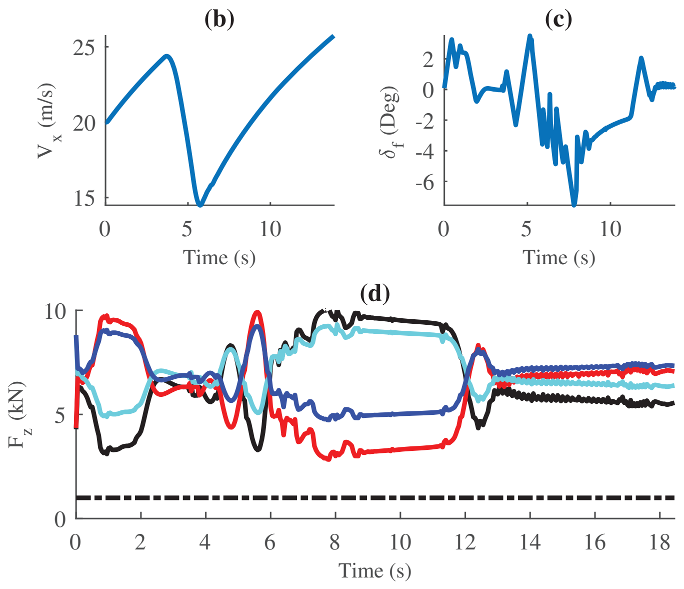


Figure 3.21: simulation-1 results for moving obstacle analysis: (b) and (c) Speed and Steering commands generated by MPC for avoiding both moving and stationary obstacles (d) Wheel lift-off constraints to meet the safety requirements.

detects moving obstacle and algorithm reroute the path, that would avoid all three moving obstacles by encompassing them under the vehicle state predictions is shown in frames 5 to 9. Here, during the frame-4, the moving obstacle state predictions are invisible due to its low prediction sampling time and low speed. i.e., the obstacle predictions are not defined for the fixed distance instead they are defined based on the prediction time stamp used in the MPC formulation. This logic has already been explained in previous section.3.3. In the frames 5 to 9, the top moving obstacles are moving downwards and bottom moving obstacle is moving upwards. However, all these moving obstacles are clearly out of the vehicle rerouted optimal path and therefore, vehicle reached final destination using this rerouted path as shown in Fig. 3.20(a). Nonetheless, the vehicle ensures vehicle dynamical safety at all times during the navigation by keeping vertical load on all tires more than specified value is shown in Fig. 3.20(d). The corresponding steering angle and vehicle speed is provided in Fig. 3.20(b) and Fig. 3.20(c).

3.5.2.2 Simulation-2 results

This section of results is extension to the previous moving obstacle section-1 and here more complicated obstacle field is added to verify the algorithm performance for

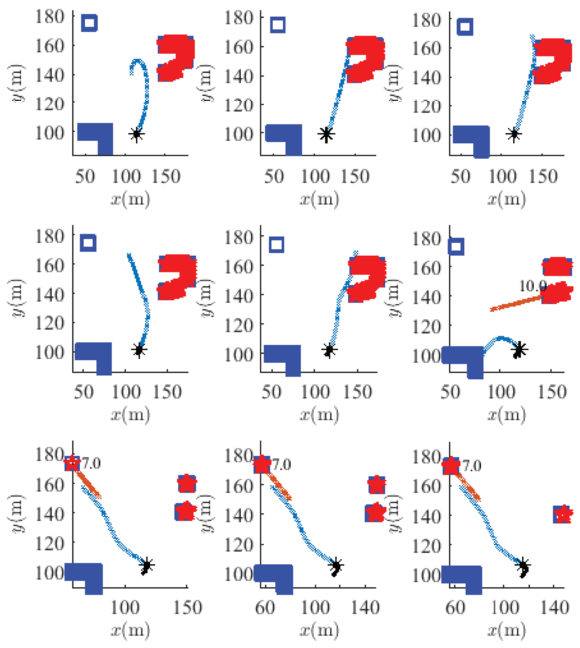


Figure 3.22: Series of nine sequential events in moving obstacle simulation-1 results for observing the first major change in AGV path; here, each event/frame represent one MPC iteration/sample period and is numbered from top left to bottom right.

moving obstacle avoidance with difficult situations. The stationary obstacle field is similar to the scenario-2 in section-2 and the moving obstacle states are considered as same as previous section given in Table. 3.3. Therefore, the vehicle should follow the optimal path given Fig. 3.16(a). However, from Fig. 3.24(a), the major path change has taken at about 4.2 sec and 4.8 sec due to the interference of moving obstacles and the corresponding detailed analysis have been provided in following sections. Fig. 3.26. is made for capturing the initial major change from the stationary obstacle optimal path by analyzing the moving obstacle states. Similar to the previous section, here the Fig. 3.26, is made with nine plots starting at 4.2 sec to observe the series of events before and after the moving obstacle interference with vehicle states. From frame 1 to 6, the algorithm detects the first moving obstacle and commands towards left to avoid collision. However, during this process, the vehicle passes through moderate steep regions by ensuring the vehicle dynamical safety. This is acceptable and necessary to meet vehicle safety constraints than passing through the steep regions. Once the vehicle is navigated out of first moving obstacle region, the algorithm commands towards optimal path by taking right turn in the subsequent iterations. However, the second obstacle from top left corner interfere with the vehicle path and maneuvers further left and this phenomenon is explained in detail in the

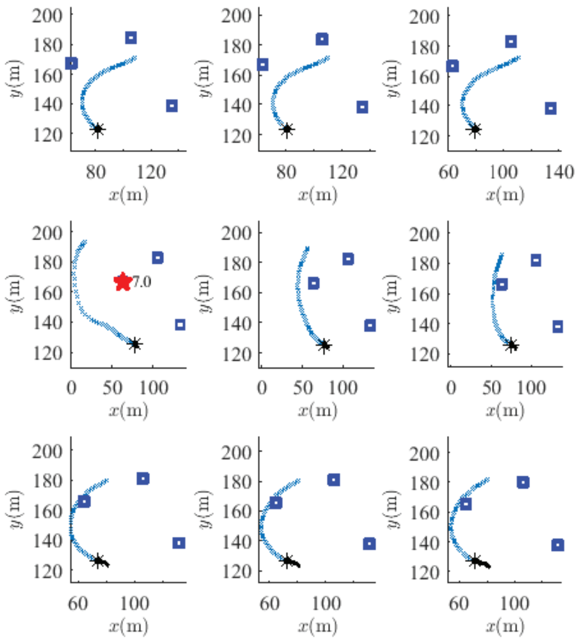


Figure 3.23: Series of nine sequential events in moving obstacle simulation-1 results for observing the second major change in AGV path.

following section. Here, the green points indicates the vehicle predictions during the moving obstacle detection process, red points indicates the detected obstacles in the LIDAR view region and blue squares indicate the undetected or hidden obstacles from vehicle path. The Fig. 3.27 shows the above mentioned second critical change in path due to moving obstacles. In the first frame, the vehicle is expected to move towards stationary optimal path to reach the target as soon as possible. However, during frame 2 to 6, the vehicle encounters with two moving obstacles that make vehicle to take further left turn to avoid collision. Here, it is important to note that, the stationary obstacles do not have state predictions and path change due to stationary obstacle is minimal compared to moving obstacle avoidance. i.e., safe region that covers for moving obstacle is lager than stationary obstacles. Therefore, the vehicle reacts to the moving obstacles from farther distance than stationary obstacles. Further, all the above scenarios are dynamically safe and collision free is shown in Fig. 3.24 (d). Finally, from the above simulation results, it is confirmed that the MPC formulation could able to successfully avoid both moving and stationary obstacles along with moderate to steep regions in the more complex obstacle fields. However, there are few exceptions and assumptions have been in the study and these are detailed in the below discussion section.

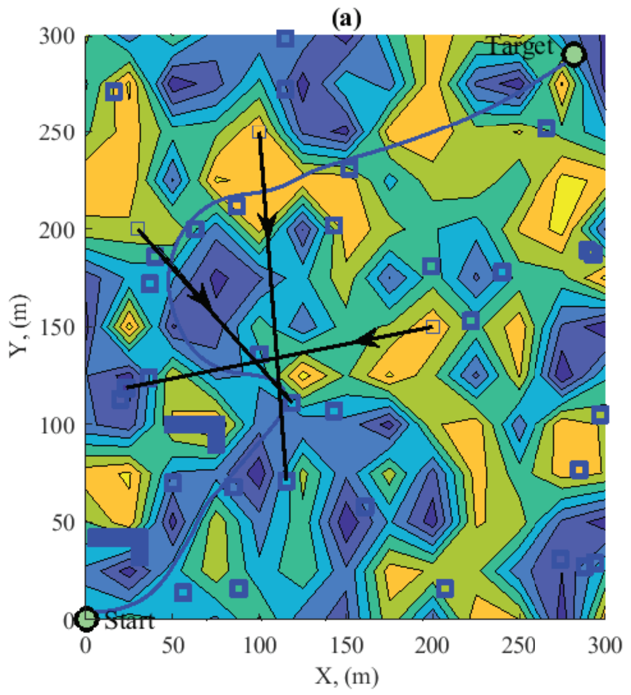


Figure 3.24: simulation-2 results for moving obstacle analysis with relatively complex obstacle field: (a) Path followed by AGV, which avoids both moving and stationary obstacles and road grade regions.

3.6 Discussion

The present study made several assumptions while developing the NMPC algorithm for navigating AGV safely and quickly. The effect of each of these assumptions and possible relaxations are explained in the following sections,

assumption-1:

In stationary obstacle detection process, it is assumed that the LIDAR can detect all obstacles in the given range. In reality, the obstacles can be with different heights and shapes and one LIDAR unit can not provide whole obstacles information for the defined LIDAR view region. Therefore, to obtain complete obstacle information, a fusion algorithm needs to be developed between multiple LIDAR sensors or a 3D LIDAR module should be used. This being said, the algorithm doe not react for the obstacles that are not recognised by the LIDAR module. Further, the LIDAR

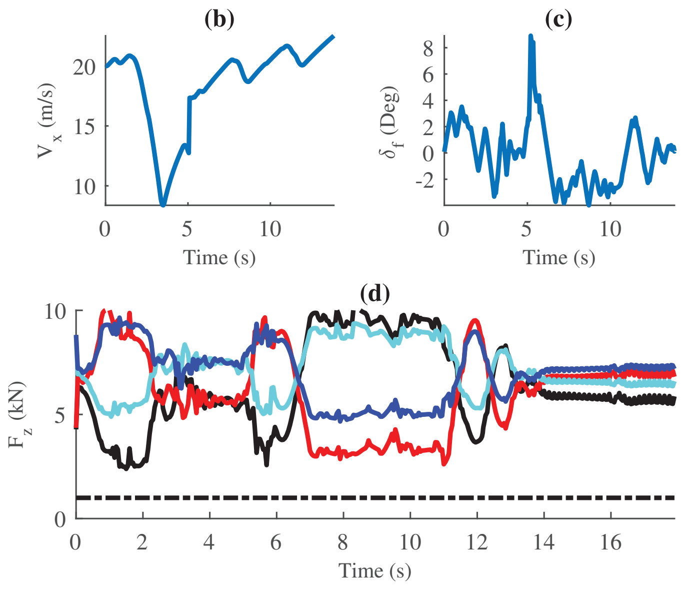


Figure 3.25: simulation-2 results for moving obstacle analysis: (b) and (c) Speed and Steering commands generated by MPC for the corresponding moving obstacle simulation-2 field scenario (d) Wheel lift-off constraints to meet the safety requirements.

module should provide obstacle states in terms of position (x_{om}, y_{om}) than distance alone. This may be possible to obtain through Leddar VU8 sensor [51], which can provide obstacle distance and angle of segment, in which the corresponding obstacle is identified.

assumption-2:

The algorithm assumes that, it obtains road grade information in terms of road grade indices in longitudinal and lateral directions for the defined LIDAR view region. The definition of road grade indices is defined in obstacle-grade avoidance process section and it is possible to obtain this data through appropriate location based GPS mapped data. The assumption to calculate θ_x and θ_y for the LIDAR view region best suits for the present simulation. However, the more detailed analysis by considering the averaged θ_x and θ_y at each node for the predicted LIDAR region and fusing these grade indices with state dynamics model would provide more control on steep regions. This increases computational burden on simulation and it is left for the further work. These GPS mapped data can be processed in off-line mode to obtain the corresponding target location terrain, soil and grade information. Therefore, with minor modifications the algorithm can easily be extended for different soil,terrain and

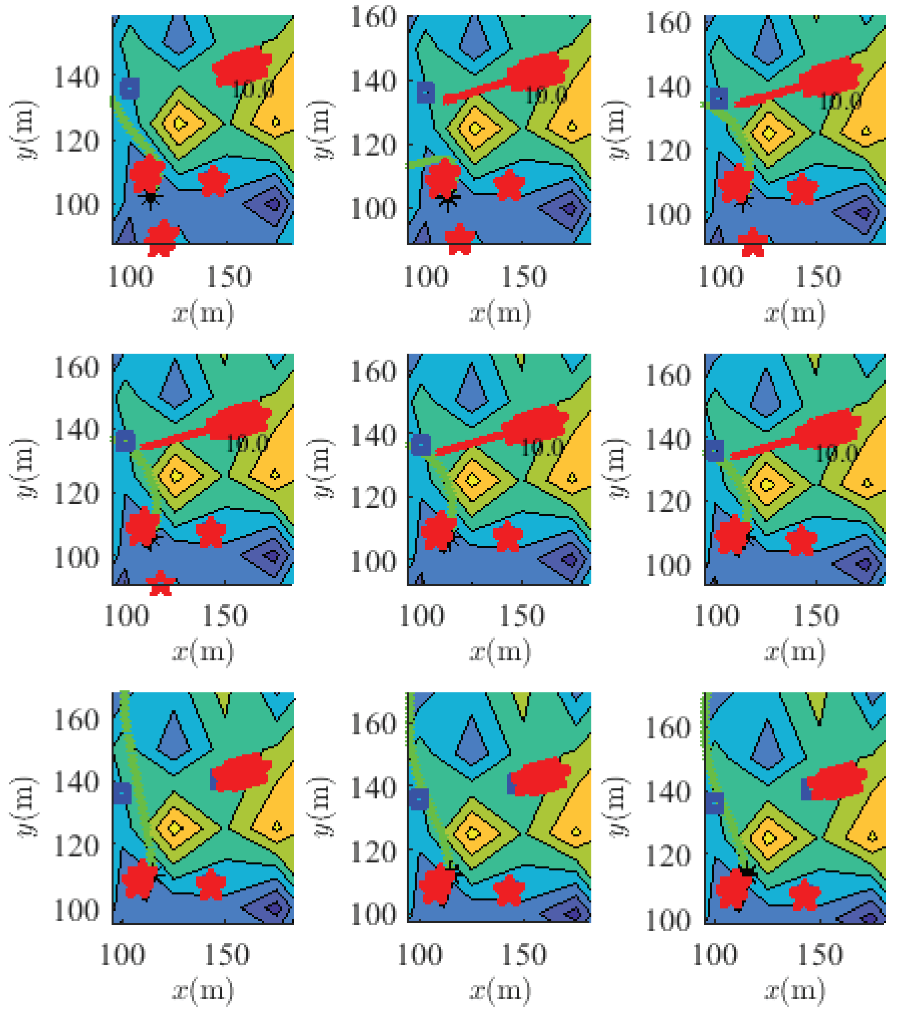


Figure 3.26: Series of nine sequential events in moving obstacle simulation-2 results for observing the first major change in AGV path; here, each event/frame represent one MPC iteration/sample period and is numbered from top left to bottom right.

weather conditions too. This left for future research work.

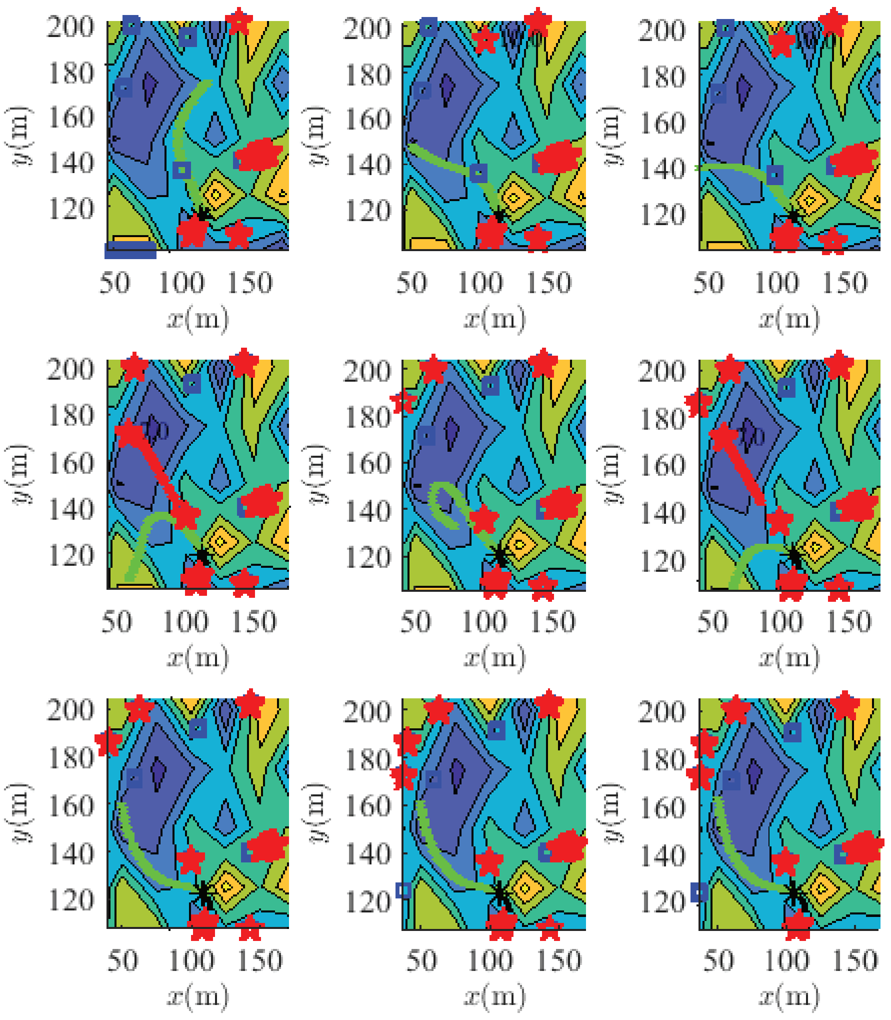


Figure 3.27: Series of nine sequential events in moving obstacle simulation-2 results for observing the second major change in AGV path.

assumption-3:

In the present study, it is assumed that the vehicle parameters are constant and its state estimations are exact. However, the developed MPC can handle considerable uncertainty in the vehicle states and the results have been demonstrated with thirty simulations in section.3.5.1.6. However, this is possible because of inherent ability of MPC controller to handle non-linear control systems and thus typical present feedback control system be handled for a limited uncertainty. This implies that, the present algorithm may not be robust enough for more significant uncertainty in the system. Further research needs to made for improving the system robustness and a robustness scheme can be incorporated to the present OCP formulation and it is left for the future work.

assumption-4:

In the present study moving obstacle detection process is made possible through the assumption that, the LIDAR sensors used for moving obstacle analysis operates n times faster than MPC iteration or sample time period. In reality, the typical update rate for LIDAR sensors can easily meet this requirement and the assumption of having n measurements in one MPC iteration is reasonable. However, there are other sensors including Radars can detect moving obstacles and provide its position, speed and angle shoot. The algorithm operates in parallel to the sensor modules and therefore the obstacle and grade map processing can be done in off-line to process its information. Further research needs to be made for training the algorithm for various obstacle fields and thus improving the algorithm speed in real time. Finally, Further, with the exception of moving obstacle analysis, the developed MATLAB code is able to execute in 0.305 sec on $2.8GHz$ intel(R) Core(TM) i5-7440 HQ processor.

Chapter 4

studies on simulation and real time implementation of LQG controller

4.1 Introduction

The demand for autonomous navigation is increasing and major car manufacturers are looking for robust and safe operation of vehicle. However, the localisation of the vehicle is a challenging factor in making robust autonomous controller. There are already some technologies exist in the market including advanced driver assistance system (ADAS) and Adaptive cruise control [57], [58]. In the present technology, the cost for measuring the accurate vehicle position has been reduced. However, the noise in sensors and handling the update rates are still challenging and require through study on each sensor. The present work consider each sensor associated with autonomous navigation and incorporate their noise properties to improve the accuracy of vehicle state estimates.
Prior studies have made for developing tracking algorithms including geometry based Stanely method [59] developed by DARPA, Pure pursuit method [60, 61, 62] and fuzzy based algorithms [63]. However, much of these methods are restricted to single input and single output systems, and thus require lot of effort in tuning gains. Further, the noise in various sensors affect the tracking performance and tuning the controller gains for each parameter would become cumbersome. However, developing model based controller can overcome these drawbacks. This is because, the model approximates the real vehicle behaviour and tuning the controller gains becomes easy [64]. Extensive work have made on these model based controllers [65, 66, 67] and effort have been made to develop accurate vehicle models including bi-cycle kinematic

and dynamic models. However, the effort on analysing the tracking characteristics including sensor noise and errors in state estimations were not considered.
The main objective in the present work is to design a full-sate Linear Quadratic Gaussian controller that would consider tracking performance and each sensor noise associated with it. Further, the detailed explanation of each sensor interface with controller is provided. During the controller development various filters are designed including low pass filter, Kalman filter. The vehicle heading values from the IMU sensor are processed to alleviate the magnetic effects on autonomous navigation and it is detailed in Chapter.2.
The remaining sections in the chapter are as follows: section 2 provide methodology for controller, section-3 provide experimental set up. This includes interfacing the various sensors with controller, vehicle actuators and test conditions used for the analysis. Section-4 provide the simulation and implementation of LQG controller on $1/5^{th}$ truck has been provided.

4.2 Methodology

The present study consider implementing LQG controller on $1/5^{th}$ truck for tracking the given target map. The target map consists of series of way points, which are required to be connected through continuous line. In the present work, cubic spline fit has been used to connect way-points. The Cubic spline fit curvature is most suitable for making real road turning paths including 90 $^\circ$ turn, round about circle and S-map path. The detailed pseudo code for developing continuous desired path explained in later sessions. In order to obtain robust controller gain (K_{gain}) and performance, it is important to mimic the real vehicle with the model equations. Further, the present work uses Kalman observer to make the best estimates from sensor measurements and vehicle model. The model developed in this study is restricted to kinematic bi-cycle model, which is sufficient for capturing the $1/5^{th}$ truck lateral dynamics is shown in Fig. 4.1. The controller outputs the steering angle that would track the path with minimum cross track error.

4.2.1 Vehicle Kinematics model

The lateral kinematics model makes following assumptions including
1. Fuses two front wheels into one single front wheel and two rear wheels into one single rear wheel.

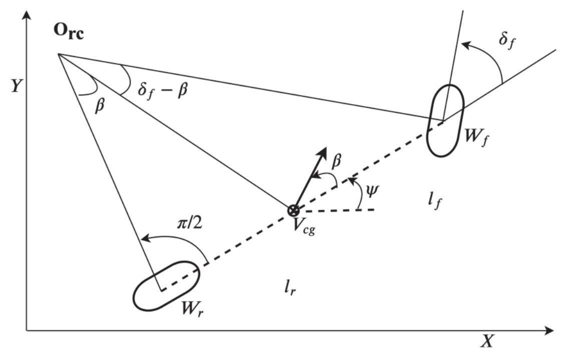


Figure 4.1: Vehicle kinematics bi-cycle model for LQG controller

2. Assuming front wheel steering alone $\delta_r = 0$
3. slip angles at both wheels are zero.

From the above assumptions and Fig .4.1, the kinematic model becomes [52]

$$\dot{X} = v_x \cos(\psi) \tag{4.1}$$

$$\dot{Y} = v_x \sin(\psi) \tag{4.2}$$

$$\dot{\psi} = \frac{V}{L} \tan(\delta_f) \tag{4.3}$$

Here v_x represents the vehicle longitudinal speed in m/s, ψ represents vehicle heading measured in global coordinates. The steering angle δ_f is further calculated from averaging the inner and outer wheel radius of front wheels [52]. From the equations 4.1- (4.3), the kinematic model equations are still in non-linear form and it is required to convert them into linear form for the controller application. Further, it is important to incorporate target path coordinates in the problem formulation for making effective path tracking controller [62]. Therefore, developing model with respect to target path and linearising it along equilibrium target points in terms of cross track error and heading error produce better tracking performance results. Fig. 4.2. provide the schematic of kinematic model in desired path coordinates and the kinematic state model with respect to equilibrium path coordinates can be written as follows,

$$\begin{bmatrix} e_{f_a} \\ \dot{e_{f_a}} \\ \theta_e \\ \dot{\theta}_e \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 0 & v_x & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_{f_a} \\ \dot{e_{f_a}} \\ \theta_e \\ \dot{\theta}_e \end{bmatrix}_{k} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_x/L \end{bmatrix} \delta_f \tag{4.4}$$

where,

$$A = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 0 & v_x & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_x/L \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

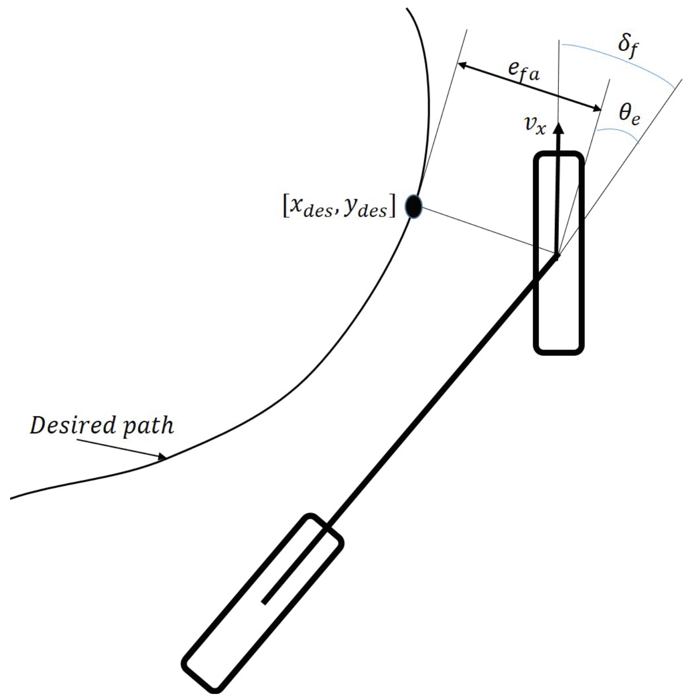


Figure 4.2: Vehicle kinematic model along desired path coordinates

Here, dt is time step for each iteration e_{f_a} is cross track error in $meters$, θ_e is heading error in rad, $\dot{e_{f_a}}$ is rate of change of cross track error in m/s and $\dot{\theta}_e$ represents rate of change of heading error with respect to the equilibrium target point.

4.2.2 LQG controller operation

The Fig.4.3. shows the schematic of LQG architecture and it includes Kalman filter for observing the vehicle states and target map in terms of waypoints is given as input to the system. The target waypoints are processed using Cubic spline before entering into the controller loop. Further, the Hardware unit in the schematic has been detailed with the fritzing circuit diagram is shown in Fig. 4.4. While developing the LQG controller, it is important to check the stability of the system model and it can be verified through the rank of observability and controllability matrices. The

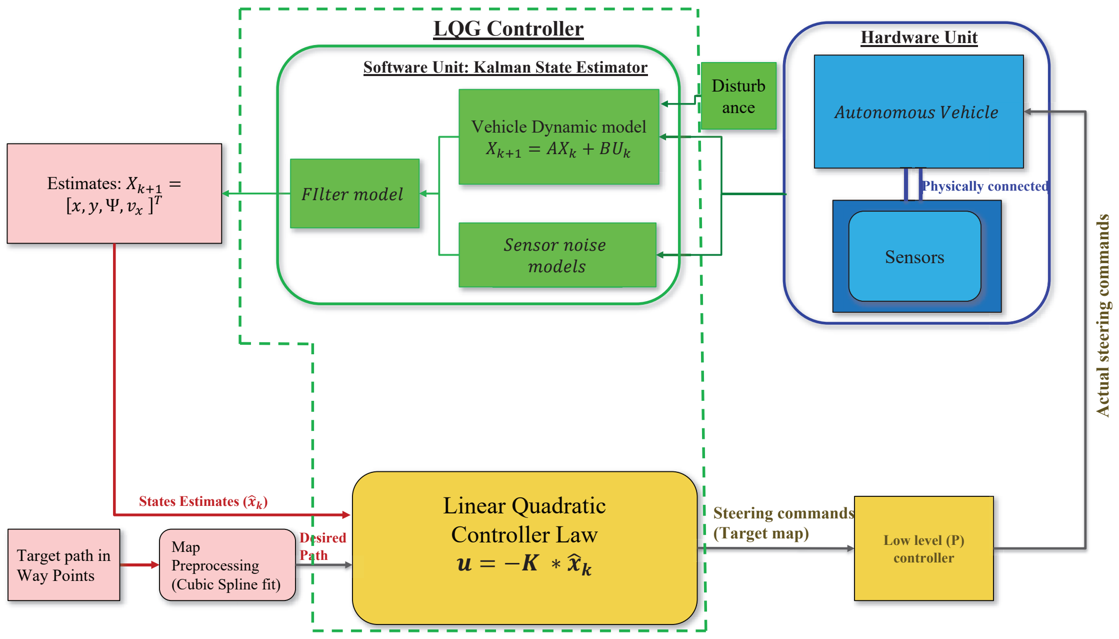

LQG Controller
Software Unit: Kalman State Estimator
Vehicle Dynamic model $X_{k+1} = AX_k + BU_k$
FIlter model
Sensor noise models
Disturbance
Hardware Unit
Autonomous Vehicle
Physically connected
Sensors
Actual steering commands
Estimates: $X_{k+1} = [x, y, \Psi, v_x]^T$
States Estimates ($\hat{x}_k$)
Target path in Way Points
Map Preprocessing (Cubic Spline fit)
Desired Path
Linear Quadratic Controller Law $\boldsymbol{u} = -\boldsymbol{K} * \hat{\boldsymbol{x}}_{\boldsymbol{k}}$
Steering commands (Target map)
Low level (P) controller

Figure 4.3: Schematic of LQG controller architecture

observability and controllability matrices can be defined as follows,

$$Controllability = \begin{bmatrix} A & AB & A^2B & . & . & A^{n-1}B \end{bmatrix} \tag{4.5}$$

$$Observability = \begin{bmatrix} C \\ CA \\ CA^2 \\ . \\ . \\ CA^{n-1} \end{bmatrix} \tag{4.6}$$

Here, the rank of each matrix is four and matches with size of states (n) in the above problem to make sure the developed model is fully controllable and observable. Once the controllability and observability have been verified, the LQR quadratic cost function along with constraints using control inputs and states can be written as,

$$\min_{\mathcal{Z},\mathcal{U}} \quad J = \mathcal{Z}^T.Q.\mathcal{Z} + \mathcal{U}^T.R.\mathcal{U} \tag{4.7}$$

s.t.

$$\dot{\mathcal{Z}} = \mathcal{D}[\mathcal{Z},\mathcal{U}] \tag{4.8}$$

$$\mathcal{Z}_{min}(t) \leq \mathcal{Z}(t) \leq \mathcal{Z}_{max}(t) \tag{4.9}$$

$$\mathcal{U}_{min}(t) \leq \mathcal{U}(t) \leq \mathcal{U}_{max}(t) \tag{4.10}$$

$$\mathcal{E}[\mathcal{Z}(t), \mathcal{Z}_{target}] \leq 0 \tag{4.11}$$

Here, $\mathcal{Z}$ represents the system states and $\mathcal{U}$ indicates the system control inputs. Further, the equation (4.11) provide the end constraint for stopping the vehicle, when it reached within the region of final waypoint and equation 4.8 represents the system model given in the above section.4.2.1. The above cost function along with constraints can be solved using Discrete Algebraic-Racatti Equation (DARE) in the following manner,

$$K_{gain} = (B^-1.Z_{DARE}.B + R)^{-1}(B^T.Z_{DARE}.A) \tag{4.12}$$

Here, Z_{DARE} is calculated as with a convergence loop for the maximum number of iterations In the present study, the vehicle has been equipped with GPS base

Algorithm 3 Solving Discrete time Algebraic Raccati Equation

```
for $i = 1 : max\ number\ of\ iterations$ do
    $Z_{DARE_{i+1}} = A^T.Z_{DARE_i}.A - (A^T.Z_{DARE_i}.B)(R + B^T Z_{DARE_i} B)(B^T Z_{DARE_i} A) + Q$

    if $abs(Z_{DARE_{i+1}} - Z_{DARE_i}) \leq Convergence_{tolerance}$ then
        break
    end if
end for
```

station, IMU, speed, and steering wheel sensors to read vehicle states. The below experimental set up section provide detailed explanation on each sensor interface with Embedded system. Further, the controller interface with vehicle actuators including DC powertrain motor and steering servo unit is also explained.

4.3 Experimental set-up

The schematic of experimental set up with hardware unit used in the path tracking analysis is shown in Fig.4.4. In the present work, $1/5^{th}$ scale truck has been used for implementing the controller. The following sections provide detailed explanation on interfacing the vehicle with sensors and embedded systems.

4.3.1 position measurements set-up

In the present work, the Inertial Sense EVB-2 board has been used for measuring the vehicle position. This module has two GPS units, in which one acts like rover and other unit acts like base station. Both units are communicated each other by on

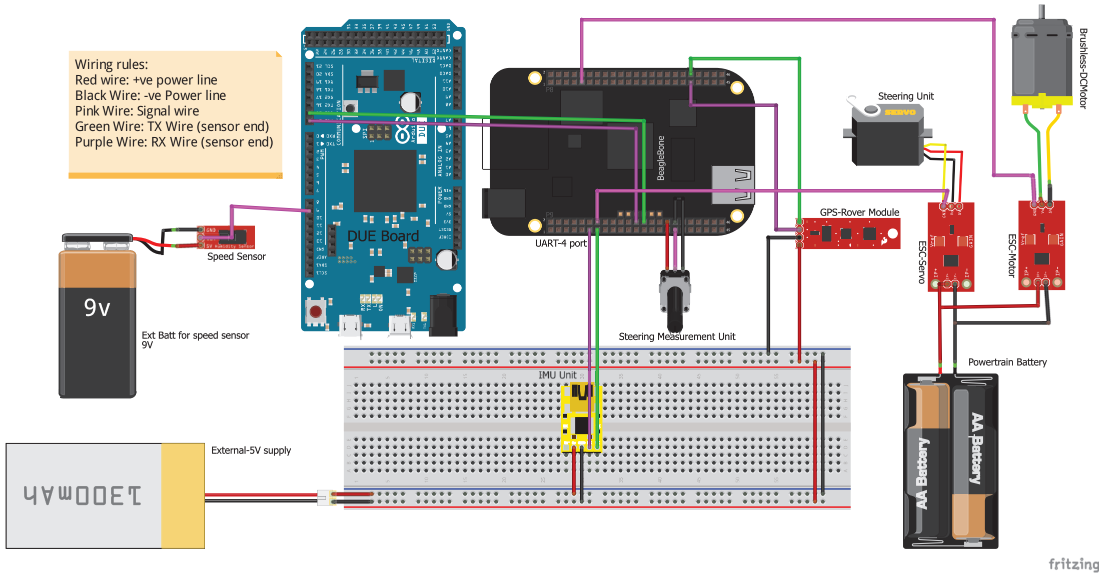


Figure 4.4: Fritzing circuit diagram for sensor interface with Beagle Bone Black Embedded system

board 915MHz X-bee radio for RTK (Real Time Kinematics) corrections. Here, the base station unit be mounted at fixed location and rover unit be mounted on vehicle. The rover unit communicates with base station through above mentioned radio for receiving the real-time position corrections from the base station unit is shown in Fig.4.5.

The present EVB-2 board provide the position measurements within the error of $\pm 0.35m$ for the given location. However, when the distance between rover and base station goes beyond 2.5km, the accuracy in position varies due to the limitation on radio communication. Therefore, for all of the experiments, it is made sure that, the GPS base station is fixed within the region of 2.5km is shown in Fig .4.5. Further, the rover unit be connected to Embedded system through serial communication.The embedded system used in the present work is a 1.2 GHz Linux based Wireless Beagle Bone Black RevC board and it has various GPIO interface pins for communicating with sensors. The Embedded system has 96 pins with four UART serial pins for communicating with GPS and IMU sensors. Here, IMU provides vehicle heading measurements and are explained in the following section.

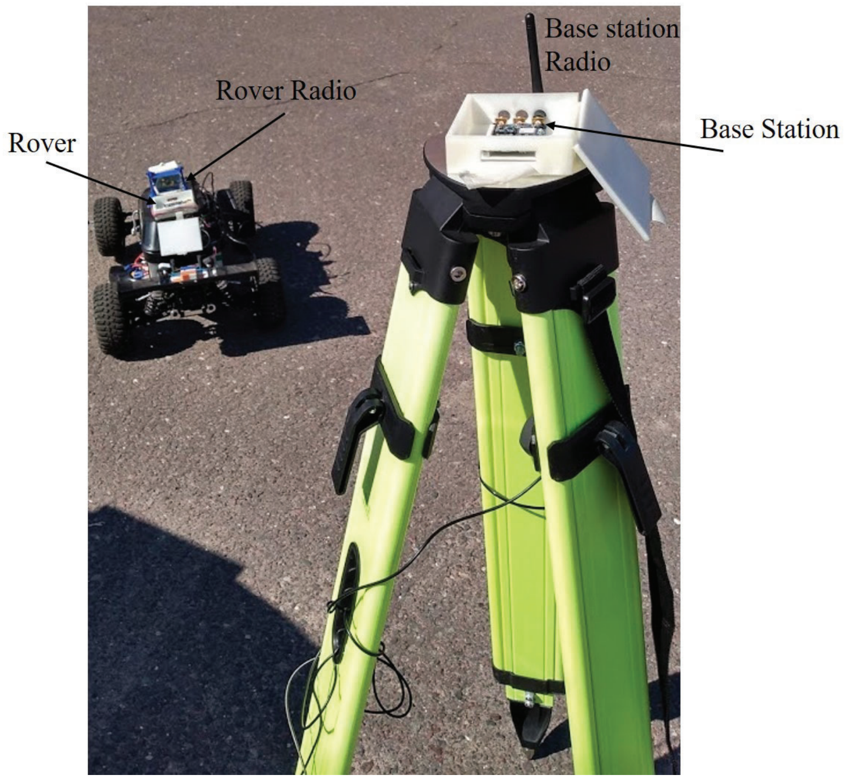


Figure 4.5: Radio communication between GPS base station and rover

4.3.2 Vehicle heading measurements

The $UM7$ inertial measurement Unit (IMU) has been used for estimating the vehicle orientation in global coordinates. The module consists of three axis gyro, accelerometer and magnetometer for providing the vehicle roll,yaw and pitch. However, the vehicle orientation can either be represented in Euler angles or Quaternions. In the present study, Quaternions have been used to estimate the vehicle orientation and to avoid the gimbol lock [16]. Similar to GPS module, the IMU interface with embedded system using UART serial pins. The update rate for the module is 80 hz and for the brevity, the vehicle heading estimations from quaternions is provided with the following calculations,

$$heading = \arctan 2(numerator, denominator) \tag{4.13}$$

Here,

$$numerator = 2(q_0.q_3 + q_1.q_2)$$
$$denominator = q_0^2 + q_1^2 - q_2^2 - q_3^2$$

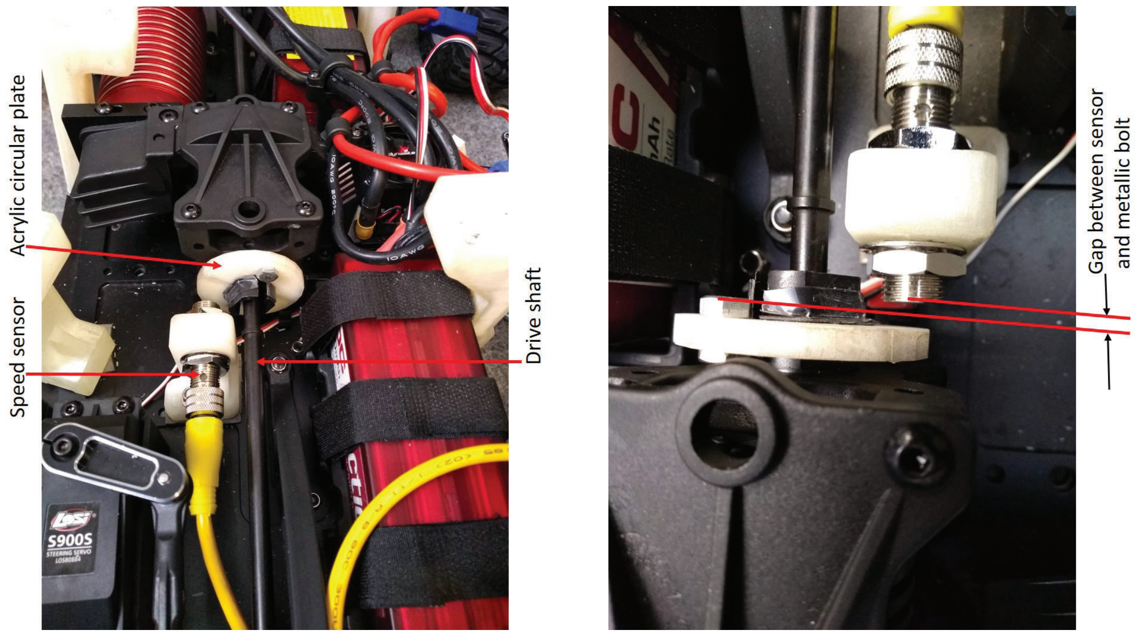


Figure 4.6: Speed sensor mounting on 1/5th truck

More details on Quaternions (q_0, q_1, q_2, q_3) and fusion of vehicle kinematics for alleviating the magnetic effects on heading is given in Chapter.2. This way, the vehicle heading and position measurements are made using base station and IMU sensors and corresponding sensor properties have been provided in the Appendix.A .

4.3.3 Vehicle speed and steering measurements

The speed measurement set up consists of three major parts including Metallic object proximity switch sensor, acrylic circular plate and metallic bolts. The bolts are attached to the circular plate at a known radius and the circular plate center is mounted on the vehicle drive shaft. The metallic speed sensor is fixed on the vehicle chassis on a certain height that would face the metallic bolt on acrylic plate is shown in Fig. 4.6. The speed sensor sense the bolt when it acrylic plate is rotated and the time period between each sensor output value provide speed measurement. The bolt comes within a distance of 1-2.5 mm with speed sensor to get recognised and the time duration between each bolt be measured with separate embedded system (Due) to process speed data. The processed speed data from Due is communicated onto Beagle bone black through UART serial communication. The baud rate used for the communication is 115200 and a fixed gear ratio has been used for converting the drive shaft speed to wheel speed is given in equation 4.14.

$$v_x = factor * \omega_{ds} \tag{4.14}$$

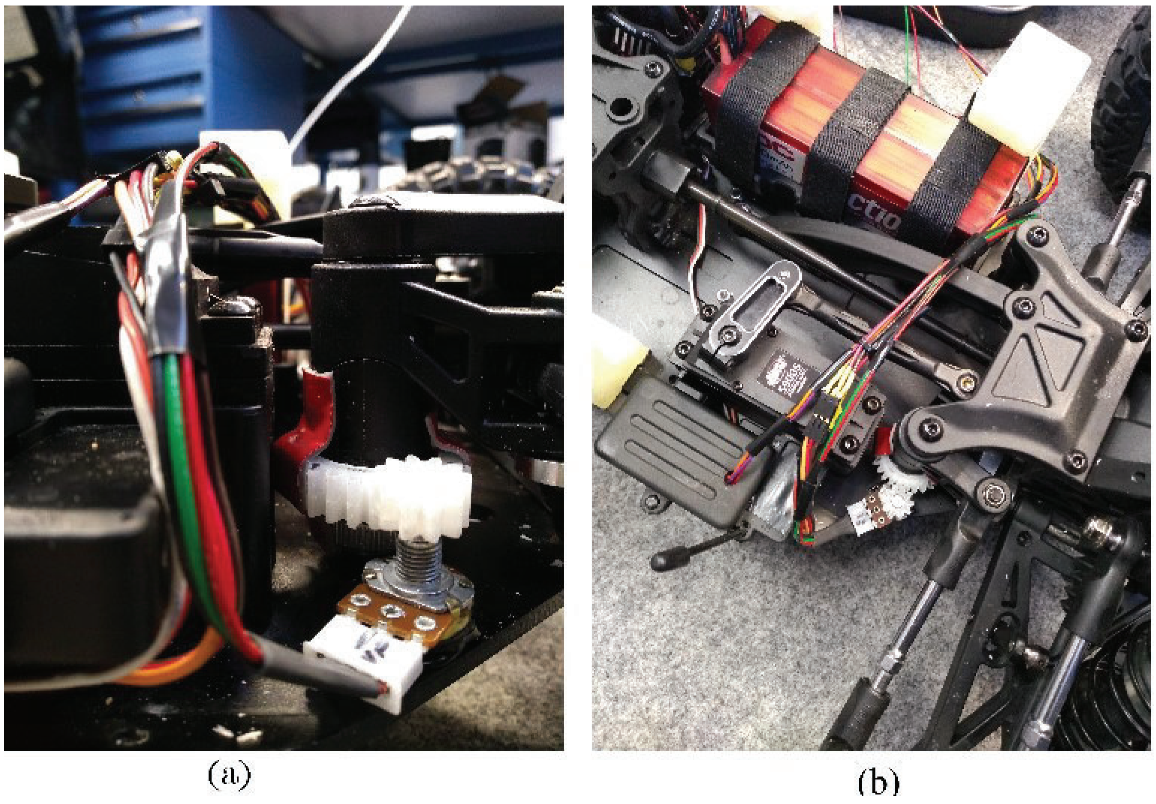


Figure 4.7: Steering sensor mounting on $1/5^{th}$ truck (Here, Speed sensor was not yet installed on vehicle)

Here, the bolt is mounted on the acrylic circular plate at a diameter of $14.25mm$, vehicle wheel diameter is $190mm$ and the ratio factor between wheel to drive shaft is calculated as 0.55. The measured vehicle speed has the accuracy of $\pm 0.05m/s$ and this data would be used with steering angle to model the vehicle states. Therefore, it is important to measure these quantities with more accuracy to fuse the model state estimations with measurements from GPS and IMU. The Kalman filter estimator has been used for fusing this information and further discussion on fusion algorithm is given in Kalman filter section.

The steering angle measurement is made with the use of potentiometer and a 3D printed gear unit that are meshed together to read the steering angle position. The 3D printed gear is attached to the steering mechanism rod and getting rotated, when the steering system is rotated is shown in Fig 4.7. The motion be transmitted into voltage through potentiometer and the calibration studies have been made to convert the steering angle into road wheel angle for the given vehicle. Here, the potentiometer is connected to one of the inbuilt analog pin on embedded system. The below equation.4.15. provide the conversion between potentiometer voltage to road wheel angle

$$RWA = 41.25 \times Volt - 19.25 \tag{4.15}$$

4.3.4 Control interface with vehicle actuators

As mentioned, the LQG controller makes the decisions based on the sensor inputs and target map. The controller provide steering angle and speed commands to the vehicle. However, the controller needs to be interfaced with motor and steering servo to implement the commands and navigate the vehicle. The vehicle has come with inbuilt Electronic speed controller (ESC) with the remote controller. However, the PWM frequency and voltage levels of ESC have been debugged using oscilloscope and function generator. The following frequency and voltage levels have been found for the steering and DC motor unit of $1/5^{th}$ truck,
Servo actuation definition:

- † Frequency of operation: $181.2Hz$
- † Peak to Peak voltage level: $3.27V$
- † Full left PWM duty cycle: 21; full rigt PWM Duty cycle: 31

DC Motor actuation definition:

- † Frequency of operation: $90.9Hz$
- † Peak to Peak voltage level: $3.27V$
- † Forward direction: stall duty cycle: 11, initialize duty cycle: 13.93
- † minimum PWM duty cycle for very low speeds: 14; maximum PWM duty cycle for top speed: 15
- † Reverse direction: stall duty cycle: 13.93, initialize duty cycle: 13.2
- † minimum PWM duty cycle for very low speeds: 12; maximum PWM duty cycle for top speed: 9

The signal pins for steering servo and DC motor from ESC are bypassed from remote controller through PWM pins on Beagle bone black. As mentioned, python has been used for implementing the PWM duty cycle onto signal pins.

4.3.5 State observer design

As mentioned, the Kalman filter is used as an observer in providing full state estimation for the LQG controller. Further, from the above explained sensor measurements, the embedded system receives data from each sensor at different update rates and it is required to obtain data with a minimum update rate of 15 hz to navigate the vehicle autonomously. However, the GPS module only update its measurements at every 0.2 sec and not sufficient for the application. The Kalman estimator consider the knowledge of noise in model development and noise in measurements and provide the best estimates using probabilistic rules at a specified time period.
The Kalman filter fusion algorithm for making vehicle position and heading estimates is made through following two steps,

4.3.5.1 Prediction step:

$$\overline{z}_{k+1} = F\hat{z}_k + Bu_k \tag{4.16}$$

$$\overline{P}_{k+1} = F\hat{P}_k F^T + Q_k \tag{4.17}$$

Where,

$$F = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ B1 \\ 0 \\ B2 \\ 0 \\ B3 \end{bmatrix}$$

$$B1 = v_x \cos\theta, B2 = v_x \sin\theta$$

$$B3 = \frac{v_x}{L} \tan\delta_f$$

Here, equation (4.16) and (4.17) represents the priori estimations at $k+1$ step, based on the vehicle kinematics and posterior estimations $(\hat{z}, \hat{P})$ at step k. Further, the error or noise in the prior estimations can be accounted through process noise Q. In the present work, for simplicity, the process noise is considered as piece wise for the implementation.

4.3.5.2 Update step

The update equations as follows,

$$\hat{z}_{k+1} = \overline{z}_{k+1} + K_{mat}.y_{mat} \tag{4.18}$$

$$\hat{P}_{k+1} = (I_{3\times 3} - K_{mat}.H_{mat})\overline{P}_{k+1} \tag{4.19}$$

Where,

$$S_{mat} = H_{mat}.\overline{P}.H_{mat}^T + R$$

$$K_{mat} = \frac{\overline{P}.H_{mat}^T}{S_{mat}}$$

$$y_{mat} = meas_{mat} - (H_{mat}.\overline{z})$$

Here, the measurement matrix consists of measurements from GPS base station and IMU sensor values. Further, the noise in measurements would be accounted in R matrix and the weightage for measurements and model predictions would be varied based on the number of satellites availability and norm of the magnetic field at the current position. Therefore, the adaptive weights would be used based on the knowledge of sensor operation.
Therefore, the

4.3.6 Test conditions

The present work includes four types of test conditions including general map, 90° turn map, round about map and inverted S-type map is shown in Fig 4.9. Here, general map indicates the path made near APS labs in Michigan Tech University. The path surrounded with several ferrous material including a dyno and vehicle could able to reject the external magnetic field while making the vehicle heading estimations. Further, the vehicle has been tested during the snow conditions too. The second map represents 90° turn to mimic the left turn on real road condition. The inverted S-type map mimics the agricultural field and 180° turn map mimics the round-about turn is shown in Figures from 4.12 to 4.10. As mentioned, the target map is made by number of way points that are apart approximately 2 m from each other. These way points are connected with smooth curve using the Cubic spline fit. The following Pseudo code provide the algorithm for Cubic spline fit that makes smooth map between the given waypoints,
The application of Cubic spline algorithm for different routes resulted in smooth desired path for LQG controller is shown in Fig. 4.8. This is because, in real time, the gps location points are not continuous and the controller require continuous line

Algorithm 4 Desired path/map creation using Cubic Spline fit

```
ds = 0.1,h[1] = 0.0, s[1] = 0.0
$A = 0_{N\times N}, A[0,0] = 1.0$
$B = 0_{N\times 1}$
for $i = 2 : no\ of\ waypoints(N) - 1$ do
    $dx[i] = x_{wp}[i+1] - x_{wp}[i]$
    $dy[i] = y_{wp}[i+1] - y_{wp}[i]$
    $s[i+1] = s[i] + \sqrt{dx[i]^2 + dy[i]^2}$
    h[i] = s[i+1]-s[i]
end for
for $k = 2 : no\ of\ waypoints(N)$ do
    if $k\ = (no\ of\ waypoints(N) - 1)$ then
        $A[k,k] = 2(h[k-1] + h[k])$
    end if
    $A[k,k-1] = h[k], A[k-1,i] = h[k]$
end for
$A[N,N-1] = 0.0, A[N,N] = 1.0$
for $k = 2 : no\ of\ waypoints(N) - 1$ do
    $B_x[k] = \frac{3(x[k+1]-x[k])}{h[k]} - \frac{3(x[k]-x[k-1])}{h[k-1]}$
    $B_y[k] = \frac{3(y[k+1]-y[k])}{h[k]} - \frac{3(y[k]-y[k-1])}{h[k-1]}$
end for
$c_x = solve(A, B_x)$
$c_y = solve(A, B_y)$
for $k = 2 : no\ of\ waypoints(N)$ do
    $d_x = \frac{(c_x[k]-c_x[k-1])}{(3.h[k-1]}, d_y = \frac{(c_y[k]-c_y[k-1])}{(3.h[k-1]}$
    $b_x = \frac{(x[k]-x[k-1])}{(h[k-1]} - \frac{h[k-1](c_x[k]+2.c_x[k-1])}{3}$
    $b_y = \frac{(y[k]-y[k-1])}{(h[k-1]} - \frac{h[k-1](c_y[k]+2.c_y[k-1])}{3}$
end for
$s_{sp} = array(0, s[end], step = 0.1)$
for $j = 1 : (no\ of\ waypoints(N))/ds$ do
    $k = bisect\ of\ ds\ (or)\ indices\ between\ two\ points$
    $ds_x = s_sp[j] - x[k]$
    $x_{spline}[j] = x[k] + b_x[k].ds_x + c_x[k].ds_x^2 + d_x[k].ds_x^3$
    $ds_y = s_sp[j] - y[k]$
    $y_{spline}[j] = y[k] + b_y[k].ds_y + c_y[k].ds_y^2 + d_y[k].ds_y^3$
    $dx_{sp} = b_x[k] + 2c_x[k].ds_x + 3.d_x[k].ds_x^2$
    $dy_{sp} = b_y[k] + 2c_y[k].ds_y + 3.d_y[k].ds_y^2$
    $Yaw_{sp} = atan2(dy, dx)$
    $ddx_{sp} = 2.c_x[k] + 6.d_x[k].ds_x$
    $ddy_{sp} = 2.c_y[k] + 6.d_y[k].ds_y$
    $Curvature_{sp} = (ddy.dx - ddx.dy)/(dx^2 + dy^2)^(1.5)$
end for
```

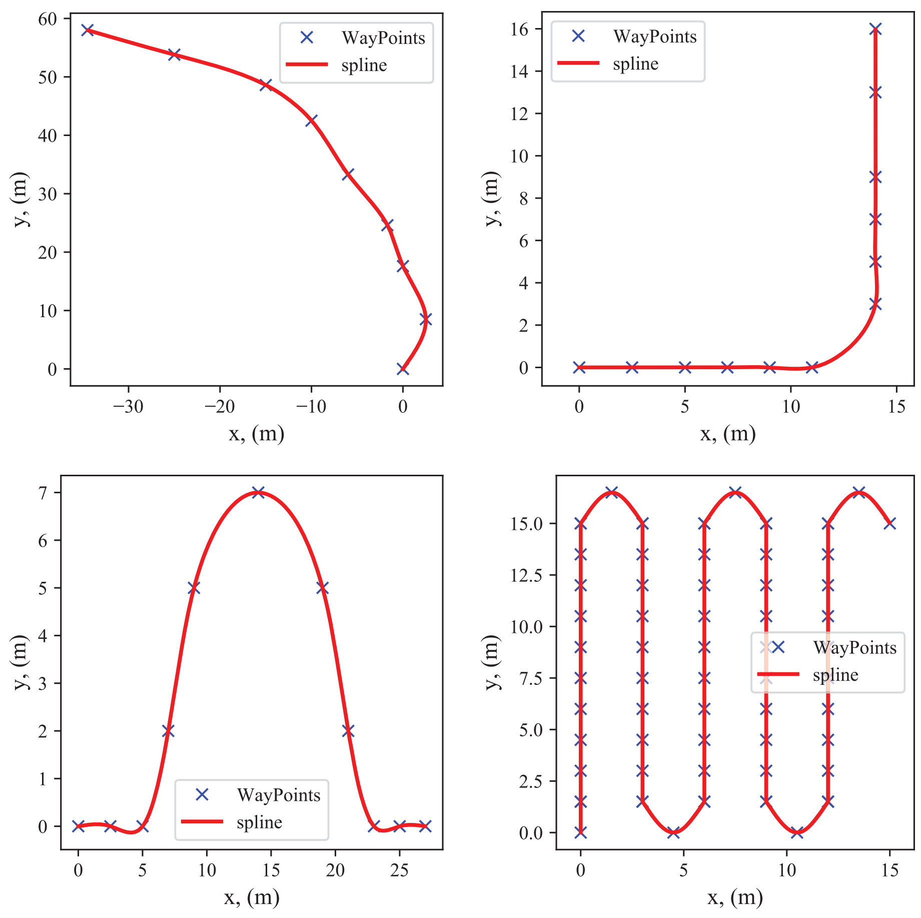


Figure 4.8: Desired path map creation using Cubic spline fit.

to track the path accurately. These desired maps from cubic spline fit closely mimic the real road conditions with the given sparse waypoints. Terefore, the developed LQG controller can be applied for real road scenarios.

4.4 Results and Discussion

The results section is mainly divided into two parts including simulation results and validation results for LQG controller. Simulations are helpful in developing the map prepossessing, controller gain tuning and analysing cross track errors before being applied onto real vehicle. This would save time and cost while developing the controller for various curvature paths.

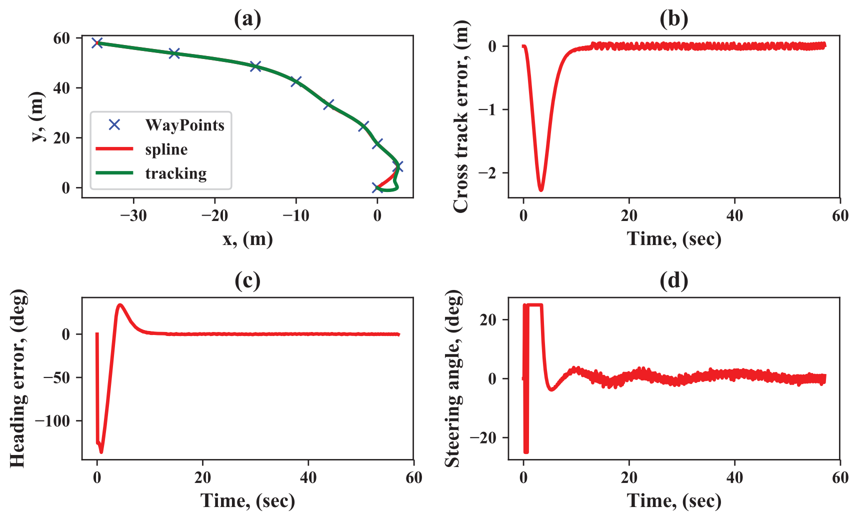


Figure 4.9: Path tracking for general APS labs path

4.4.1 Simulation results

4.4.1.1 simulation-1 results:AV-4 map results

The simulations are made using Python script and each subsystem considered as method in the code. As mentioned, the results were made for various curvature paths that would mimic the real road conditions. From Fig. 4.9, it can be clearly seen that, the controller is able to track the given AV-4 map within the cross track error of $\pm 0.25m$ error. Here, the waypoints are indicated with x symbol, and red curve shows the Cubic spline that connects the waypoints to formulate desired path. The green line shows the simulation tracking result, that tries to follow the desired path. However, at the initial hick up, the vehicle tries to move away from the path and once the controller gets stabilized, the controller was able to track the path within $\pm 0.2m$. This could be because of Kalman filter observer matrix initialization and it requires some time period for stabilizing the filter propagation and update steps. However, during the actual tests, this initialization has been taken care by iterating the filter before entering into the path tracking loop.

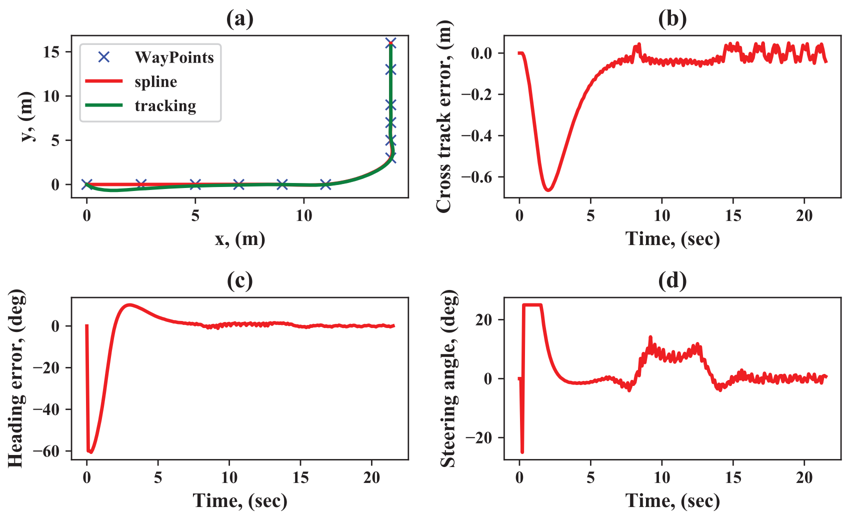


Figure 4.10: Path tracking for left-turn map

4.4.1.2 simulation-2 results:90° turn map

These tests mimic the left turn on real road conditions. Similar to above AV-4 map results. the left turn results shows the initial hick up at the starting of path and slowly merge with desired path with the maximum cross track error of $\pm 0.2m$ is shown in Fig.4.10. Here, the desired path at 90 $^\circ$ turn is made with Cubic spline fit and the curvature at this turn made the controller to follow the desired path smoothly. The Fig.4.10(d) shows the steering angle command for the given path and it has been restricted between its maximum limits of $\pm 20°$. The heading error along the path shows that, the vehicle is able to orient its heading along the path and reduces unnecessary deviations from the desired path.

4.4.1.3 simulation-3 results:Round-about map

The round about map is one of the difficult paths that vehicle has to make while vehicle is on the road. From Fig.4.11, it is observed that, the vehicle is able to track the desired path within the error of $\pm 0.2m$. Similar to left turn map, the cubic spline made the desired path curvature smooth and the points were connected by meeting the continuity and curvature conditions at the waypoints.

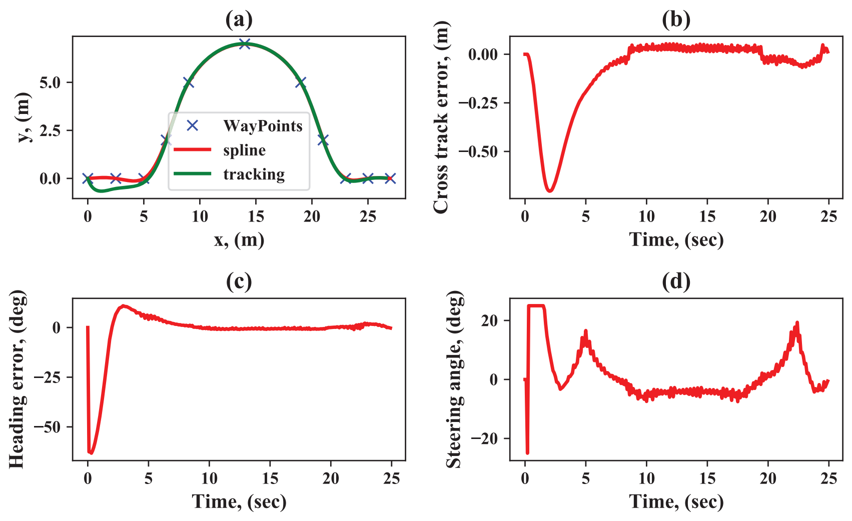


Figure 4.11: Path tracking for Round-about map

4.4.1.4 simulation-4 results:AF map

These simulations mimic the typical field conditions, where the vehicle makes continuous inverted 'S' loops over and over for the entire field. The Fig.4.12 shows the tracking performance of LQG controller for AF path. The desired path for AF field is not smooth at the curves and thus the cross track error after the initial hick up is shooting above 0.2 m. This can further be reduced by altering and inserting more waypoints at the curve portion and thus improving the smoothness of desired path. This is one of major advantage of developing simulations, while design and implementing the controller on actual vehicle. The next section provide the validation studies, that would provide real time implementation results for $1/5^{th}$ vehicle.

4.4.2 Validation results

As mentioned, the LQG controller is implemented on $1/5^{th}$ truck with the maximum steering limits of $\pm 25°$. The heading and GPS sensor provide measurements in global coordinates and the vehicle model, controller have been developed with respect to Cartesian coordinates. Therefore, all the measurements were converted into Cartesian coordinates, before entering into controller loop. This makes the positive steering angle to take left turn and negative steering command to take right turn while vehicle is tracking the path. Further, a low level proportional controller have been implemented to track the commanded steering angle from LQG controller.

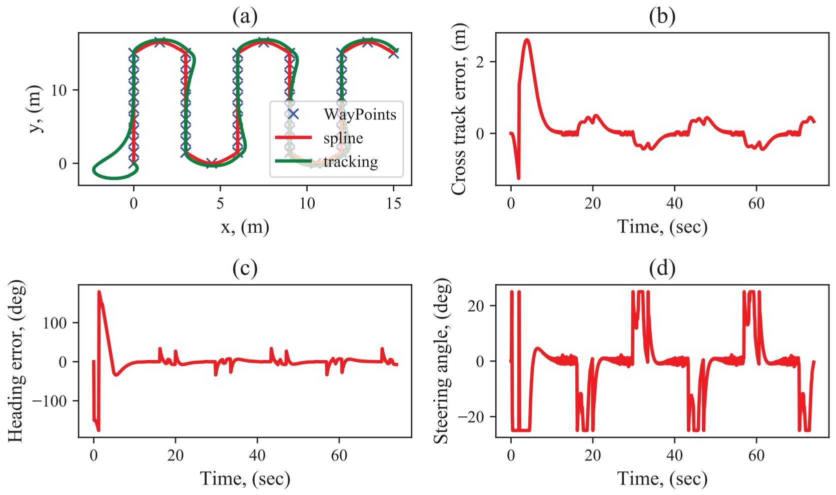


Figure 4.12: LQG path tracking results on AF map

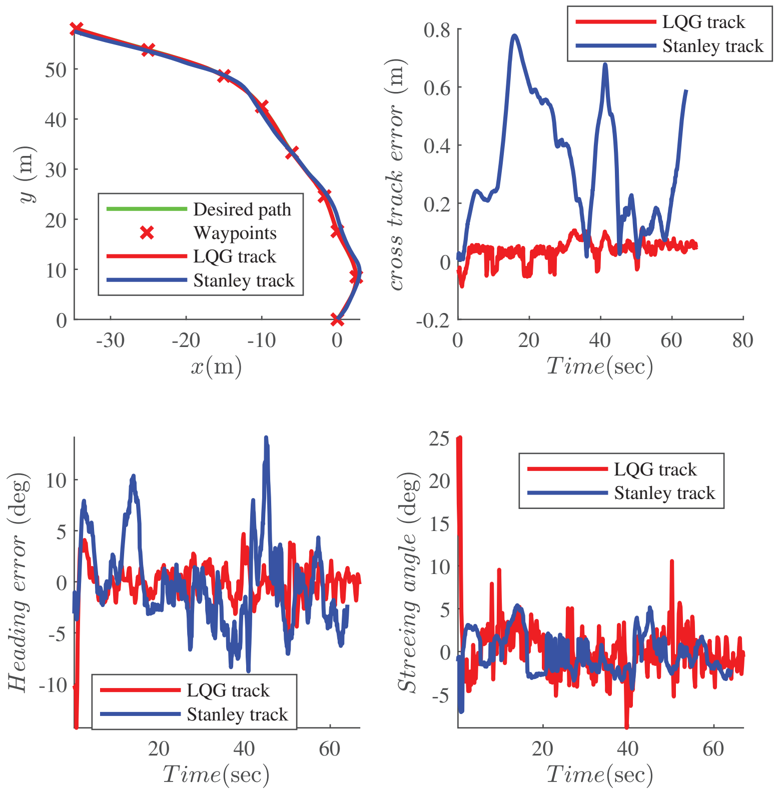


Figure 4.13: LQG path tracking test results comparison for AV-4 map

Fig.4.13 shows the tracking test results for AV-4 map using LQG controller and Stanley controller [59]. The methodology for Stanley controller is provided in Appendix. C.3. From the results, it can be clearly seen that, the vehicle is able to track the given path within $\pm 0.1m$ cross track error using LQG controller. However, the same path was tracked by using Stanley controller with same test conditions and sensor modules. The Stanley controller cross track error in Fig.4.13(b) has been shooting upto 0.8 m during the tracking and it clearly shows the advantage of having state estimation analysis in the controller development. The LQR controller has been equipped with the Kalman observer for the state estimation and further have the robust control logic compared to the simple Stanley control law. From Figures 4.9 and 4.13, it can be concluded that, the simulation results from LQG controller are in good agreement with the test results. The similar trend can be observed for the remaining drive cycles too. The development of sensor interface with beagle bone black, Kalman state estimator along with cubic spline desired path laid out the potential platform for implementing the tracking and obstacle avoidance using LQG and MPC controller and it is left for the future work. Further, these controllers can exploit the obstacle avoidance logic from chapter.3.

Chapter 5

Summary of Work

The present research work analysed various aspects of autonomous vehicle development including sensor fusion analysis, developing non-linear MPC controller algorithm for off road high speed application and real time implementation of LQG controller on $1/5^{th}$ truck. The conclusions from each study have been provided as follows,

5.1 Conclusions

A modified Extended Kalman filter has been made by fusing the vehicle kinematics information for alleviating magnetic disturbances on ground vehicle yaw estimations. The filter reacts to the amount of external magnetic disturbances and replaces the horizontal magnetic field vector with vehicle kinematics information. Further, based on steering wheel angle and vehicle speed, the modified EKF can make accurate yaw estimations both in straight line and turning conditions. The results showed that the modified EKF has improved the performance of filter in magnetic disturbance environment. With this modified EKF approach, while the vehicle is affected by 1 ± 0.8 Norm magnetic field it can reduce the maximum RMS errors in heading estimations from 3.4 to 0.5° in straight path and 6.0 to 1.9 ° during tuning paths. Due to high accuracy in speed sensor and steering angle measurements, this fusion algorithm works for long distances and can sustain for longer periods without much drift in heading estimations. Further work can be conducted to improve heading estimates by fusing the GPS heading information and adding vehicle dynamics into

the algorithm. This is left for the future work.
The study of sensor fusion has been provided in the above in section and below section extends the conclusions and summary on non-linear MPC controller algorithm.
The Non-linear MPC controller has been developed for off-road high speed AGVs application that would avoid both stationary and moving obstacles by meeting the vehicle dynamical safety constraints and yet reaches the target location as soon as possible. Further, the vehicle tries to avoid steep regions while maintaining minimum specified vertical load on each tire. A new algorithm called ABD-JDN algorithm is developed based on box slope and box detect methods to process the obstacles information and CasADi tool is used to fuse the moving obstacles states information into N-MPC problem formulation. This framework provides strong communication between obstacle movement predictions and vehicle future predictions and prepares the vehicle to take diversion in well advance from dangerous zone. Further, thirty simulations are made by adding noise of $\pm 0.35m$ in vehicle states, $\pm 2.5^{\circ}$ in vehicle heading and $\pm 0.05m$ in obstacle distance measurements. The algorithm provided robust performance for the considered vehicle states and obstacle states uncertainty limits and all thirty simulation runs reached target location as soon as possible with collision free and yet ensured the vehicle dynamical safety for the entire path. Finally, the simple and efficient lidar process for handling stationary obstacle-grade fields along with the symbolic problem formulation using CasADi tool would execute the control algorithm such that the real time implementation on vehicle can be achieved.
The above two sections provide the summary on fusion and control algorithm development studies. However, the final work presents the conclusions on real time implementation of LQG controller by exploiting the developed algorithms from sensor fusion and N-MPC controller.
The sensor fusion analysis provide robust heading data to the controller and vehicle is stable enough to deal with external magnetic fields. The N-MPC model development provided a way to formulate the problem and possible potential future obstacle avoidance implementations can be made. However, the present work is implemented the path tracking on $1/5^{th}$ for various paths including left turn, AV-4 path, round-about path and AF-path using LQG controller. Here, the given target map is made with the set of way points, that are connected using Cubic spline curve. Therefore, the path curvature is also calculated for the analysis. The fusion algorithm makes the best estimates by fusing the GPS base station position with vehicle kinematics model. The tracking results show that the vehicle could able track the path with the lateral error of $\pm 0.25m$ error and experimental results matches well with the simulation results.

5.2 Discussion

This section provide discussion on making connection between three research works conducted in the present study. As mentioned, the first phase of current study concentrates on handling different sensors for providing accurate vehicle state estimates. The Kalman filter fusion algorithms developed in the study are used in various parts of LQG controller development including position, yaw, velocity and steering angle estimations. For instance, while implementing the LQG controller on $1/5^{th}$ truck without the fusion algorithm for AV-4 path, the vehicle was wobbling near the dyno region. This is because, the magnetometer was getting affected by the external magnetic field and leading to erroneous vehicle yaw measurements. This has been eliminated by using the developed Modified EKF algorithm in the LQG controller and processed the IMU sensor reading for accurate yaw estimations. Similar to the yaw estimations, the fusion algorithm has also been applied to position and velocity measurements and improved controller tracking performance from $\pm 0.6m$ to $\pm 0.2m$. This clearly shows the systematic fusion algorithms reduces the noise in vehicle state estimations and improve controller performance. This way, the developed algorithms in first phase of research are linked with third phase of LQG controller implementation work.
Similarly, the second phase of research is concentrated on developing algorithms for off road autonomous vehicles that can avoid both stationary and moving obstacles. The results are generated using MATLAB based simulations on $2.8GHz$ intel(R) Core(TM) i5-7440 HQ processor, and it is concluded that, the average time for each MPC iteration is taking upto 0.305 sec. This includes the number of predictions (N) of 100, stored obstacle information for upto 106 and inclusion of longitudinal vehicle dynamics in the problem formulation. However, by reducing the number of predictions and number of stored obstacles would reduce the processing time to approximately $1/4^{th}$ times. Further, the generation of C-code from MATLAB functions and CasADi tools reduces the parameter memory allocation and improves efficiency of the code. This makes real time implementation of these algorithms along with sensor fusion is possible and it is left for the future work. However, the methodology used in the second phase of research is replicated in LQG controller implementation. From the above conclusions and discussions, it is clear that, the base platform has been laid out for developing and implementing MPC controller for both path tracking and obstacle avoidance and yet reach the target location as soon as possible and this is left for the future work. and explained in the future work section.

Chapter 6

Future Work

The future work from the present work can be summarised as follows:

1. The present work, fusion of vehicle kinematics with IMU magnetometer for alleviating the magnetic effects on vehicle yaw estimations have been studied. However, incorporating the base station GPS position states would further improve the fusion algorithm and thus robustness to external magnetic fields.
2. In the non-linear MPC controller development for grading avoidance, the value of grade at each node can be evaluated and fused into the vehicle dynamic state equation for further improving the grade performance of controller. However, the computational burden would be increased appropriately.
3. Currently, the non-linear MPC controller is able reach target location with considerable noise in the vehicle parameters. However, the model may diverge with the additional noise in the parameters and it is recommended to develop a robustness scheme to appropriately handle the noise factors.
4. The real time implementation of developed non-linear MPC controller is a challenging task. This may be achieved by offloading some of the calculations either in offline or through using multi-threading operation. These calculations may include terrain map processing, obstacle processing for reducing the load on main controller.
5. The mentioned non-linear MPC algorithm can be trained for different terrain maps, obstacle fields and grading conditions such that the calibration look up tables can be generated. This task can be achieved through the appropriate machine learning algorithms that can reduce computational burden for real time implementation.

6. The LQG controller implementation along with obstacle avoidance and vehicle safety constraints can be incorporated to the existing work. Further, the noise in obstacle distance measurements can be incorporated to improve the vehicle safety in obstacle avoidance.

References

[1] T. Litman, *Autonomous vehicle implementation predictions.* Victoria Transport Policy Institute Victoria, Canada, 2017.

[2] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 3, pp. 143–153, 2003.

[3] D. F. Llorca, V. Milanés, I. P. Alonso, M. Gavilán, I. G. Daza, J. Pérez, and M. Á. Sotelo, "Autonomous pedestrian collision avoidance using a fuzzy steering controller," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 390–401, 2011.

[4] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *IEEE transactions on intelligent transportation systems*, vol. 7, no. 1, pp. 20–37, 2006.

[5] M. Kok, J. D. Hol, T. B. Schön, F. Gustafsson, and H. Luinge, "Calibration of a magnetometer in combination with inertial sensors," in *2012 15th International Conference on Information Fusion.* IEEE, 2012, pp. 787–793.

[6] M. Pettersson, "Extended kalman filter for robust uav attitude estimation," 2015.

[7] Z.-Q. Zhang, X.-L. Meng, and J.-K. Wu, "Quaternion-based kalman filter with vector selection for accurate orientation tracking," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 10, pp. 2817–2824, 2012.

[8] G. Wahba, "A least squares estimate of satellite attitude," *SIAM review*, vol. 7, no. 3, pp. 409–409, 1965.

[9] R. G. Valenti, I. Dryanovski, and J. Xiao, "A linear kalman filter for marg orientation estimation using the algebraic quaternion algorithm," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 2, pp. 467–481, 2015.

[10] X. Yun, E. R. Bachmann, and R. B. McGhee, “A simplified quaternion-based algorithm for orientation estimation from earth gravity and magnetic field measurements,” *IEEE Transactions on instrumentation and measurement*, vol. 57, no. 3, pp. 638–650, 2008.

[11] R. Mahony, T. Hamel, and J.-M. Pflimlin, “Nonlinear complementary filters on the special orthogonal group,” *IEEE Transactions on automatic control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[12] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, “Estimation of imu and marg orientation using a gradient descent algorithm,” in *2011 IEEE international conference on rehabilitation robotics*. IEEE, 2011, pp. 1–7.

[13] F. L. Markley, “Attitude error representations for kalman filtering,” *Journal of guidance, control, and dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

[14] J. L. Crassidis and F. L. Markley, “Unscented filtering for spacecraft attitude estimation,” *Journal of guidance, control, and dynamics*, vol. 26, no. 4, pp. 536–542, 2003.

[15] D. Choukroun, I. Y. Bar-Itzhack, and Y. Oshman, “Novel quaternion kalman filter,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 174–190, 2006.

[16] C.-R. U.-I. manuals, “An-1008-sensors for orientation estimation,” in *Document rev. 1.0, Oct. 2012, pp. 1-11, “AN-1005-Undestanding Euler Angles”, Document rev. 1.1*. IEEE, Mar 2013, pp. 1–12.

[17] R. Costanzi, F. Fanelli, N. Monni, A. Ridolfi, and B. Allotta, “An attitude estimation algorithm for mobile robots under unknown magnetic disturbances,” *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 4, pp. 1900–1911, 2016.

[18] X. Tong, Z. Li, G. Han, N. Liu, Y. Su, J. Ning, and F. Yang, “Adaptive ekf based on hmm recognizer for attitude estimation using mems marg sensors,” *IEEE Sensors Journal*, vol. 18, no. 8, pp. 3299–3310, 2017.

[19] K. Feng, J. Li, X. Zhang, C. Shen, Y. Bi, T. Zheng, and J. Liu, “Correction: A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm. sensors 2017, 17, 2146,” *Sensors*, vol. 17, no. 11, p. 2530, 2017.

[20] Y. S. Suh, Y. S. Ro, and H. J. Kang, “Quaternion-based indirect kalman filter discarding pitch and roll information contained in magnetic sensors,” *IEEE Transactions on Instrumentation and measurement*, vol. 61, no. 6, pp. 1786–1792, 2012.

[21] G. Shi, X. Li, and Z. Jiang, "An improved yaw estimation algorithm for land vehicles using marg sensors," *Sensors*, vol. 18, no. 10, p. 3251, 2018.

[22] R. Rajamani, *Vehicle dynamics and control.* Springer Science & Business Media, 2011.

[23] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[24] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments," *Vehicle System Dynamics*, vol. 56, no. 6, pp. 853–882, 2018. [Online]. Available: https://doi.org/10.1080/00423114.2017.1399209

[25] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, Oct 2017.

[26] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986. [Online]. Available: https://doi.org/10.1177/027836498600500106

[27] S. Shimoda, Y. Kuroda, and K. Iagnemma, "High-speed navigation of unmanned ground vehicles on uneven terrain using potential fields," *Robotica*, vol. 25, no. 4, p. 409–424, 2007.

[28] B. D. Luders, S. Karaman, and J. P. How, *Robust Sampling-based Motion Planning with Asymptotic Optimality Guarantees.* [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2013-5097

[29] A. Hussein, H. Mostafa, M. Badrel-din, O. Sultan, and A. Khamis, "Metaheuristic optimization approach to mobile robot path planning," in *2012 International Conference on Engineering and Technology (ICET)*, Oct 2012, pp. 1–6.

[30] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *2013 European Control Conference (ECC)*, July 2013, pp. 4136–4141.

[31] P. Ogren and N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, April 2005.

[32] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, “Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads,” in *ASME 2010 dynamic systems and control conference*. American Society of Mechanical Engineers Digital Collection, pp. 265–272.

[33] Y. Rasekhipour, A. Khajepour, S. Chen, and B. Litkouhi, “A potential field-based model predictive path-planning controller for autonomous road vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[34] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Birkhäuser, 2012, vol. 26.

[35] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[36] J. Park, D.-W. Kim, Y. Yoon, H. J. Kim, and K. Yi, “Obstacle avoidance of autonomous vehicles based on model predictive control,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, pp. 1499 – 1516, 2009.

[37] H. O. J. BEVAN, G. P; GOLLEE, “Trajectory generation for road vehicle obstacle avoidance using convex optimization,” *Proceedings of the Institution of Mechanical Engineers. Part D, Journal of automobile engineering*, 2010.

[38] M. Werling and D. Liccardo, “Automatic collision avoidance using model-predictive online optimization,” in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 6309–6314.

[39] R. Attia, J. Daniel, J. Lauffenburger, R. Orjuela, and M. Basset, “Reference generation and control strategy for automated vehicle guidance,” in *2012 IEEE Intelligent Vehicles Symposium*, June 2012, pp. 389–394.

[40] J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg, “Receding horizon maneuver generation for automated highway driving,” *Control Engineering Practice*, vol. 41, pp. 124 – 133, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066115000726

[41] S. M. Erlien, S. Fujita, and J. C. Gerdes, “Shared steering control using safe envelopes for obstacle avoidance and vehicle stability,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 441–451, Feb 2016.

[42] M. Seder and I. Petrovic, “Dynamic window based approach to mobile robot motion control in the presence of moving obstacles,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 1986–1991.

[43] T. Dewi, N. Uchiyama, and S. Sano, "Service mobile robot control for tracking a moving object with collision avoidance," in *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, June 2015, pp. 1–6.

[44] B. Damas and J. Santos-Victor, "Avoiding moving obstacles: the forbidden velocity map," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 4393–4398.

[45] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "An mpc algorithm with combined speed and steering control for obstacle avoidance in autonomous ground vehicles," in *ASME 2015 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection.

[46] J. Liu, P. Jayakumar, J. Stein, and T. Ersal, "A multi-stage optimization formulation for mpc-based obstacle avoidance in autonomous vehicles using a lidar sensor," vol. 2, 10 2014.

[47] T. Shim and C. Ghike, "Understanding the limitations of different vehicle models for roll dynamics studies," *Vehicle System Dynamics*, vol. 45, no. 3, pp. 191–216, 2007. [Online]. Available: https://doi.org/10.1080/00423110600882449

[48] D. Assanis, Z. Filipi, S. Gravante, X. Gui, L. Louca, D. Rideout, J. Stein, and Y. Wang, "Validation and use of simulink integrated, high fidelity, engine-in-vehicle simulation of the international class vi truck," 03 2000.

[49] T. Ersal, M. Brudnak, A. Salvi, J. L. Stein, Z. Filipi, and H. K. Fathy, "Development and model-based transparency analysis of an internet-distributed hardware-in-the-loop simulation platform," *Mechatronics*, vol. 21, no. 1, pp. 22 – 29, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957415810001388

[50] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," in *SAE International Congress and Exposition*. SAE International, feb 1987. [Online]. Available: https://doi.org/10.4271/870421

[51] LeddarTechInc., *LeddarVu and configurator User Guide*, P/N 54A0028-2, 01202017 © 2017.

[52] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[53] R. A. Bixel, G. J. Heydinger, D. A. Guenther, and S. J. Novak, "Sprung/unsprung mass properties determination without vehicle diassembly," SAE Technical Paper, Tech. Rep., 1996.

[54] S. Rakheja and A. Piche, "Development of directional stability criteria for an early warning safety device," *SAE transactions*, pp. 877–889, 1990.

[55] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[56] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[57] N. Lyu, C. Deng, L. Xie, C. Wu, and Z. Duan, "A field operational test in china: Exploring the effect of an advanced driver assistance system on driving performance and braking behavior," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 65, pp. 730 – 747, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1369847816306672

[58] M. Kyriakidis, C. van de Weijer, B. van Arem, and R. Happee, "The deployment of advanced driver assistance systems in europe," *Available at SSRN 2559034*, 2015.

[59] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," in *The 2005 DARPA grand challenge.* Springer, 2007, pp. 1–43.

[60] O. Amidi and C. E. Thorpe, "Integrated mobile robot control," in *Mobile Robots V*, vol. 1388. International Society for Optics and Photonics, 1991, pp. 504–523.

[61] M.-W. Park, S.-W. Lee, and W.-Y. Han, "Development of lateral control system for autonomous vehicle based on adaptive pure pursuit algorithm," in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014).* IEEE, 2014, pp. 1443–1447.

[62] M. Elbanhawi, M. Simic, and R. Jazar, "Receding horizon lateral vehicle control for pure pursuit path tracking," *Journal of Vibration and Control*, vol. 24, no. 3, pp. 619–642, 2018.

[63] S. Dixit, S. Fallah, U. Montanaro, M. Dianati, A. Stevens, F. Mccullough, and A. Mouzakitis, "Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects," *Annual Reviews in Control*, vol. 45, pp. 76–86, 2018.

[64] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2015, pp. 1094–1099.

[65] Y. Xia, F. Pu, S. Li, and Y. Gao, "Lateral path tracking control of autonomous land vehicle based on adrc and differential flatness," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3091–3099, 2016.

[66] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles," *Control Engineering Practice*, vol. 61, pp. 307–316, 2017.

[67] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multi-constraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2016.

[68] S. Yim, K. Jeon, and K. Yi, "An investigation into vehicle rollover prevention by coordinated control of active anti-roll bar and electronic stability program," *International Journal of Control, Automation and Systems*, vol. 10, no. 2, pp. 275–287, 2012.

[69] Y. S. Suh, "Orientation estimation using a quaternion-based indirect kalman filter with adaptive estimation of external acceleration," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 12, pp. 3296–3305, 2010.

[70] H. Pacejka, *Tire and vehicle dynamics.* Elsevier, 2005.

[71] A. Rucco, G. Notarstefano, and J. Hauser, "Optimal control based dynamics exploration of a rigid car with longitudinal load transfer," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1070–1077, 2013.

[72] J. M. Snider *et al.*, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[73] K. Lee, S. Jeon, H. Kim, and D. Kum, "Optimal path tracking control of autonomous vehicle: Adaptive full-state linear quadratic gaussian (lqg) control," *IEEE Access*, vol. 7, pp. 109 120–109 133, 2019.

[74] S. Jeon, K. Lee, H. Kim, and D. Kum, "Path tracking control of autonomous vehicles using augmented lqg with curvature disturbance model," *International Conference on Control Robot System Society*, pp. 1543–1548, 2019.

Appendix A

Vehicle orientation representation using Quaternions

A.1 Quaternion properties and rotations

Quaternion definition: Quaternion consists of 4 components, with 1 scalar and 3 vectors. Further, the 3 vectors or imaginary components represents x, y and z axes respectively. The Quaternion (q) can be written as,

$$q = q_0 + q_1\bar{i} + q_2\bar{j} + q_3\bar{k} \tag{A.1}$$

Here,q_0, represents the scalar part and q_1, q_2 and q_3 represents magnitudes in x, y and z axes respectively. Alternatively, Quaternion can be represented in matrix form as,

$$q = \begin{bmatrix} q0 \\ q1 \\ q2 \\ q3 \end{bmatrix} = \begin{bmatrix} q_0 \\ q_v \end{bmatrix} \tag{A.2}$$

Quaternion Conjugate:

$$q^* = q_0 - q_1\bar{i} - q_2\bar{j} - q_3\bar{k} \tag{A.3}$$

Quaternion addition and subtraction: When two quaternions p and q are added together it becomes,

$$p + q = \begin{bmatrix} p_0 + q_0 \\ p_1 + q_1 \\ p_2 + q_2 \\ p_3 + q_3 \end{bmatrix}, p - q = \begin{bmatrix} p_0 - q_0 \\ p_1 - q_1 \\ p_2 - q_2 \\ p_3 - q_3 \end{bmatrix} \tag{A.4}$$

Quaternion multiplication: Quaternion multiplication is not commutative and it has following results of multiplying two vectors,

$$\begin{aligned} ij &= k, ji = -k \\ jk &= i, kj = -i \\ ki &= j, ik = -j \\ ii = jj &= kk = -1 \end{aligned}$$

Therefore, multiplication of Quaternions p and q becomes,

$$q \otimes p = \begin{bmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_1p_0 + q_0p_1 - q_3p_2 + q_2p_3 \\ q_2p_0 + q_3p_1 + q_0p_2 - q_1p_3 \\ q_3p_0 - q_2p_1 + q_1p_2 + q_0p_3 \end{bmatrix} \tag{A.5}$$

The symbol $\otimes$ indicates Quaternion multiplication. Further, the above can be written in matrix multiplication form as follows,

$$q \otimes p = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{A.6}$$

Unit norm property: Every Quaternion has to satisfy its unit-norm property called,

$$|q|^2 = |q_0|^2 + |q_v|^2 = 1 \tag{A.7}$$

Therefore, for any angle θ, $\cos^2\theta + \sin^2\theta$=1 holds true. Since, $q_0^2 \leq 1$, There must exist angle θ such that,

$$\begin{aligned} \cos^2\theta &= q_0^2 \\ \sin^2\theta &= |q_v|^2 \end{aligned}$$

Under the assumption of $-\pi < \theta < \pi$, the association of angle θ with Quaternion becomes,

$$q = \cos\theta + u\sin\theta \tag{A.8}$$

Where, u is a unit vector and θ represents the amount of rotation around an axis, which is defined by the unit vector u. **Quaternion rotation:**
For the given unit Quaternion, $q = q_0 + q_v = \cos\theta + u\sin\theta$, the operation $q \otimes r \otimes q^*$ represents rotation of vector r in **3D** space with an angle of 2θ about q_v as the axis

of rotation. Therefore, the rotation operation becomes,

$$r' = q \otimes r \otimes q^* \tag{A.9}$$

Where, r' is the rotated matrix and q^* is Quaternion conjugate. By expanding equation A.9

$$r' = (q_0 + q_1\bar{i} + q_2\bar{j} + q_3\bar{k})(r_x\bar{i} + r_y\bar{j} + r_z\bar{k})(q_0 - q_1\bar{i} - q_2\bar{j} - q_3\bar{k}) \tag{A.10}$$

By using the above Quaternion multiplication rule in Equation A.10 it becomes,

$$r' = q \otimes r \otimes q^* = Cr \tag{A.11}$$

Where,

$$C = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

The above matrix C is called the rotation matrix and this matrix is used for rotating given vector from body frame to reference or vice versa. Further, this matrix is important in making conversion between Quaternion to Euler angle conversion too [21].

Therefore, the matrix C_r^b, which is rotation matrix from reference frame to $B-frame$ can be written as follows;

$$C_r^b = C^T = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{A.12}$$

Derivation for time propagation using gyro sensor output:

$$S(w) = \begin{bmatrix} 0 & -w_x & -w_y & -w_z \\ w_x & 0 & w_z & -w_y \\ w_y & -w_z & 0 & w_x \\ w_z & w_y & -w_x & 0 \end{bmatrix}, S(q) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix},$$

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, w = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$$

Where $S(w)$, $S(q)$ denote linear mapping from $\Re^3$ to $\Re^{(4\times4)}$ and linear mapping from $\Re^4$ to $\Re^3 \times \Re^4$ respectively [15].

System dynamics equation with gyro bias becomes,

$$\dot{q} = \frac{1}{2}S(w - b^g)q = \frac{1}{2}S(q - b^g)w$$

$$= \frac{1}{2}\begin{bmatrix} 0 & -w_x & -w_y & -w_z \\ w_x & 0 & w_z & -w_y \\ w_y & -w_z & 0 & w_x \\ w_z & w_y & -w_x & 0 \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & -b^g_x & -b^g_y & -b^g_z \\ b^g_x & 0 & b^g_z & -b^g_y \\ b^g_y & -b^g_z & 0 & b^g_x \\ w^g_z & w^g_y & -b^g_x & 0 \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (A.13)$$

The final discretized state propagation matrix can be written for the sake of clarity as,

$$q_{k+1} = \frac{1}{2}S(q_k)w - \frac{T}{2}S(q_k)b^g + q_k \quad (A.14)$$

$$b^g_{k+1} = b^g_k$$

$$x_{k+1} = Ax_k + Bu_k \quad (A.15)$$

$$\begin{bmatrix} q \\ b^g \end{bmatrix}_{k+1} = \begin{bmatrix} I_{4\times4} & -\frac{T}{2}S(q) \\ 0_{3\times4} & I_{3\times3} \end{bmatrix}_k \begin{bmatrix} q \\ b^g \end{bmatrix}_k + \begin{bmatrix} \frac{T}{2}S(q) \\ 0_{3\times3} \end{bmatrix}_k w_k \quad (A.16)$$

By expanding Equation.A.16, it becomes,

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ b^g_x \\ b^g_y \\ b^g_z \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\frac{T}{2}(-q_1) & -\frac{T}{2}(-q_2) & -\frac{T}{2}(-q_3) \\ 0 & 1 & 0 & 0 & -\frac{T}{2}(q_0) & -\frac{T}{2}(-q_3) & -\frac{T}{2}(q_2) \\ 0 & 0 & 1 & 0 & -\frac{T}{2}(q_3) & -\frac{T}{2}(q_0) & -\frac{T}{2}(-q_1) \\ 0 & 0 & 0 & 1 & -\frac{T}{2}(-q_2) & -\frac{T}{2}(-q_1) & -\frac{T}{2}(-q_0) \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ b^g_x \\ b^g_y \\ b^g_z \end{bmatrix}_k$$

$$+ \frac{T}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{k+1}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \quad (A.17)$$

A.2 Derivations for Linearization of acceleration measurement model

As mentioned, the non-linear accelerometer model is as follows,

$$\overline{y}_a^b = C_r^b(-g) + e_a^b \tag{A.18}$$

The above model can be linearized using Taylor series expansion

$$f(x)|_{x=a} = f(a) + f'(a)(x-a) + f''(a)(x-a)^2 + ... \tag{A.19}$$

This can be applied to above accelerometer model (by removing higher order terms) as follows;
The non-linear rotation matrix C_r^b for accelerometer model, can be written as f_a and therefore,

$$f_a(q_k)|_{q_k=q_{k-1}} = f_a(q_{k-1}) + f_a'(q_{k-1})(q_k - q_{k-1}) \tag{A.20}$$

From Equations. A.19 and A.20, the linearized accelerometer model becomes,

$$\overline{y}_a^b = [f_a'(q_{k-1})(-q)]q_k + [f_a(q_{k-1}) - f_a'(q_{k-1})q_{k-1}](-g) + e_a^b \tag{A.21}$$

The above Equation A.21, is in the form of $y = Cx + D$ and the extra terms D can be neglected from the equations,
Therefore, the final linearized accelerometer model becomes,

$$\overline{y}_a^b = [f_a'(q_{k-1})(-g)]q_k + e_a^b \tag{A.22}$$

Where, $f_a'(q_{k-1})$ is called Jacobian of accelerometer model and assuming no external acceleration, the gravitational vector g becomes $C1^T$. Therefore, the non-linear rotation matrix in Equation A.22 becomes,

$$f_a = f_a'(q_{k-1})(-g) = -\begin{bmatrix} 2(q_1q_3 - q_0q_2) \\ 2(q_2q_3 + q_0q_1) \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{A.23}$$

And the corresponding accelerometer model Jacobian becomes,

$$f_a'(q_{k-1}) = \frac{\partial f_a(q)}{\partial q}|_{q=q_{k-1}} = \begin{bmatrix} \frac{\partial f_{a1}}{\partial q_0} & \frac{\partial f_{a1}}{\partial q_1} & \frac{\partial f_{a1}}{\partial q_2} & \frac{\partial f_{a1}}{\partial q_3} \\ \frac{\partial f_{a2}}{\partial q_0} & \frac{\partial f_{a2}}{\partial q_1} & \frac{\partial f_{a2}}{\partial q_2} & \frac{\partial f_{a2}}{\partial q_3} \\ \frac{\partial f_{a3}}{\partial q_0} & \frac{\partial f_{a3}}{\partial q_1} & \frac{\partial f_{a3}}{\partial q_2} & \frac{\partial f_{a3}}{\partial q_3} \end{bmatrix}_{k-1}$$

$$= 2\begin{bmatrix} -q_2 & q_3 & -q_0 & q_1 \\ q_1 & q_0 & q_3 & q_2 \\ q_0 & -q_1 & -q_2 & q_3 \end{bmatrix}_{k-1}$$

Therefore, the final expanded accelerometer measurement model without external acceleration would become,

$$\bar{y}_a^b = C_a q_k + e_a^b \tag{A.24}$$

Where, Jacobian $C_a = 2 \begin{bmatrix} -q_2 & q_3 & -q_0 & q_1 \\ q_1 & q_0 & q_3 & q_2 \\ q_0 & -q_1 & -q_2 & q_3 \end{bmatrix} = f_a'(q_{k-1})$ However, with external acceleration the gravity vector would have non-zero terms in it and thus, the Jacobian (C_a) for non-zero measurement model can be written as follows,

$$\begin{aligned} [f_a'(q_{k-1}) &= C_a \\ &= 2 \begin{bmatrix} C_{a1,1}a_r & C_{a1,2}a_r & C_{a1,3}a_r & C_{a1,4}a_r \\ C_{a2,1}a_r & C_{a2,2}a_r & C_{a2,3}a_r & C_{a2,4}a_r \\ C_{a3,1}a_r & C_{a3,2}a_r & C_{a3,3}a_r & C_{a3,4}a_r \end{bmatrix}_{k-1} \end{aligned}$$

where,

$$\begin{aligned} C_{a1,1} &= \begin{bmatrix} q_0 & q_3 & -q_2 \end{bmatrix}; C_{a1,2} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}; C_{a1,3} = \begin{bmatrix} -q_2 & q_1 & -q_0 \end{bmatrix}; \\ C_{a1,4} &= \begin{bmatrix} -q_3 & q_0 & q_1 \end{bmatrix}; C_{a2,1} = \begin{bmatrix} -q_3 & q_0 & q_1 \end{bmatrix}; C_{a2,2} = \begin{bmatrix} q_2 & -q_1 & q_0 \end{bmatrix}; \\ C_{a2,3} &= \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}; C_{a2,4} = \begin{bmatrix} -q_0 & -q_3 & q_2 \end{bmatrix}; C_{a3,1} = \begin{bmatrix} q_2 & -q_1 & q_0 \end{bmatrix}; \\ C_{a3,2} &= \begin{bmatrix} q_3 & -q_0 & -q_1 \end{bmatrix}; C_{a3,3} = \begin{bmatrix} q_0 & q_3 & -q_2 \end{bmatrix}; C_{a3,4} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}; \end{aligned}$$

Where $a_r = (a_e - g)$, is called acceleration in reference frame with external acceleration (a_e).

A.3 Derivations for Linearization of magnetometer measurement model:

Similar to accelerometer measurement model, the non-linear magnetometer measurement model is,

$$\bar{y}_m^b = C_r^b(m_r) + e_m^b \tag{A.25}$$

The magnetometer measurement model can be linearized similar to above accelerometer measurement model except that the gravity vector would be replaced with reference magnetic field vector at the given location. The reference magnetic field vector is defined as $m_r = \begin{bmatrix} m_{xr} & m_{xr} & m_{xr} \end{bmatrix}^T$. Therefore, the Jacobean for magnetometer

Table A.1
mean and variance of 3-axis gyro, accelerometer and magnetometer sensor test data

Component	a_x	Component	m_z
mean	-0.006571	mean	0.772032
std	0.000695	std	0.006216
Component	a_y	Component	w_x
mean	-0.006571	mean	0.772032
std	0.000695	std	0.006216
Component	a_z	Component	w_y
mean	-0.006571	mean	0.772032
std	0.000695	std	0.006216
Component	m_x	Component	w_z
mean	-0.006571	mean	0.772032
std	0.000695	std	0.006216
Component	m_y	-	-
mean	-0.006571	-	-
std	0.000695	-	-

measurement model can be computed as follows,

$$\begin{aligned}[f'_m(q_{k-1}) &= C_m \\ &= 2\begin{bmatrix} C_{a_{1,1}}m_r & C_{a_{1,2}}m_r & C_{a_{1,3}}m_r & C_{a_{1,4}}m_r \\ C_{a_{2,1}}m_r & C_{a_{2,2}}m_r & C_{a_{2,3}}m_r & C_{a_{2,4}}m_r \\ C_{a_{3,1}}m_r & C_{a_{3,2}}m_r & C_{a_{3,3}}m_r & C_{a_{3,4}}m_r \end{bmatrix}_{k-1}\end{aligned}$$

The final magnetometer measurement model becomes,

$$\overline{y}_m^b = C_m q_k + e_m^b \tag{A.26}$$

A.4 IMU Sensor properties

Table. A.1 provide the sensor noise in IMU sensor.

A.5 Test conditions

Table A.2

Test condition matrix and corresponding Mag zones for Mag-affected tests.

Test Condition	Mag effect applied	Mag effect removed	Distance traveled in Mag zone, (m)	Total length of test condition, (m)
St. line towards West	At S1 point	At S2 point	10 m	27 m
St. line towards East	At S2 point	At S1 point	10 m	27 m
90 Deg turn East to North	At N2 point	At N1 point	Approx. 14 m	Approx. 28.7 m
90 Deg turn South to West	At N1 point	At N2 point	Approx. 14 m	Approx. 28.7 m
90 Deg turn West to North	At N3 point	At N1 point	Approx. 14 m	Approx. 27.71 m
90 Deg turn South to East	At N1 point	At N3 point	Approx. 14 m	Approx. 27.71 m
5m circle round-about towards West	At C1 point	At C2 point	Approx. 12 m	Approx. 31 m
5m circle round-about towards East	At C1 point	At C3 point	Approx. 12 m	Approx. 31 m
7m circle round-about towards West	At C4 point	At C5 point	Approx. 14.1 m	Approx. 33.3 m
7m circle round-about towards East	At C4 point	At C6 point	Approx. 14.1 m	Approx. 33.3 m
circle test in clock-wise direction	Starting of 2nd circle	End of 2nd circle	31.4 m (circumference of 5m radius circle)	Total 3 circles with 5m radius, 94.2 m

Appendix B

NMPC controller code

B.1 Stationary Obstacle avoidance code

```
%% step 1: define decision variables
clear all
close all
clc
% to add casadi path to the program
addpath('G:\My Drive\PhD_work\AV2_Project\N-MPC_stuff\↩
   casadi-windows-matlabR2016a-v3.4.5');
import casadi.*
% STEP 1: define constants and design varibales
N = 50;
dt = 0.15;
%## vehicle parameters:
mass = 2689;     % in kgs
lf = 1.58; lr = 1.72; % front,rear axle to C.G. location
Iz = 4110;       % in kg-m^2
rob_diam = 2.5;
L = lf+lr;        % wheel base
delta_f_max = pi/4; delta_f_min = -delta_f_max;
```

```
%## define control inputs
jerk = SX.sym('Jk');
str_rate = SX.sym('str_rate');
T = SX.sym('Tp');
%## define states
Pos_x = SX.sym('s1');
Pos_y = SX.sym('s2');
theta = SX.sym('theta');
vx = SX.sym('vx');
vy = SX.sym('vy');          % define y_dot
wz = SX.sym('wz');
ax = SX.sym('acce');  % define varibale long velo, vx in↩
    m/sec
delta_f = SX.sym('del_f');    % steering angle
FE_effi_Tar = SX.sym('Effi_Tar');
FE_effi_req = SX.sym('Effi_req');
states = [Pos_x;Pos_y;theta;vx;vy;wz;ax;delta_f];   % ↩
   these are our final states required for the analysis
n_states = length(states);
n_ref_states = 4;                   % x_tar,y_tar,theta_tar,↩
   velo_tar (adding velo too)
controls = [jerk;str_rate;T];        % these are control↩
    inputs
n_controls = length(controls);
FE_states = [FE_effi_Tar;FE_effi_req];
n_FE_states = length(FE_states);
% define length of vehicle safety constrints,
n_veh_safty_states = 4;
no_of_obs_detect = 106;obs_states=2;
n_obs_detected = no_of_obs_detect*obs_states;
% apply moving constraint for obstacle avoidance
n_min_dist = 1;
n_slope = 2;
n_hdng_err = (N+1);
```

```
U = SX.sym('U',n_controls , N);
P = SX.sym('P',n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope+n_hdng_err);
pred_state_matrix = SX.sym('X_pred',n_states,(N+1));
hdng_err_state = 1;
xdot = states(4)*cos(states(3)) - (states(5)+lf*states↩
   (6))*sin(states(3));
ydot = states(4)*sin(states(3)) + (states(5)+lf*states↩
   (6))*cos(states(3));
thetadot = states(6);
xdotdot = states(7);
slope= [P(n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope-1);P(n_states+n_ref_states+↩
   n_FE_states+n_obs_detected+n_slope)];
[Fyf,Fyr] = Tire_model(states(4),states(5),states(6),↩
   states(8),states(7),slope(1));
ydotdot = (Fyf+Fyr)/mass - states(4)*states(6);
thetadotdot = (Fyf*lf - Fyr*lr)/Iz;
acce_dot = controls(1);
delta_dot = controls(2);
% add hdng error through P-parameter
hdng_pred_matrix = P(n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope+1:n_states+n_ref_states+↩
   n_FE_states+n_obs_detected+n_slope+n_hdng_err);
sys_model = [xdot;ydot;thetadot;xdotdot;ydotdot;↩
   thetadotdot;acce_dot;delta_dot];
model_fn = Function('f',{states,controls,slope},{↩
   sys_model});
% STEP 2: formulate Objective function:
obj = 0;     % initialize with objective fn
g = [];      % constraint vector
% integral gains:
Q = zeros(4,4);
Q(1,1) = 0.05; Q(2,2) = 0.05; Q(3,3) = 0.00; Q(4,4)= 50;
```

```
w_hdng_err = 60; Q_delta = 0.005;
R = zeros(3,3); R(1,1) = 0.05; R(2,2) = 0.5;  R(3,3) = ↩
   0.00;
w_FE = 0.0;w_phi_f = 5e-2;w_Fz = 0.005;
% terminal cost weights
w_dist = 0.5;
w_phi = 15;
w_obs = 0.005;
w_t = 0.001;
% not yet used weight terms:
w_delta = 0.1;
w_str = 1;
w_j = 0.01;
w_cf = 1;
% vehicle safety constraints terms:
Fz_thr = 1000; % Threshold vertical load for vehicle ↩
   safety constraint, in N
Fz_off = 300;   % in N
a_term = Fz_thr + 3*Fz_off;
b_term = Fz_off;
Lidar_dist_limit = 75;
% define parameter for making equal prediction length,
n_pred_len = 1;
states_curr = pred_state_matrix(:,1);
g = [g; states_curr-P(1:n_states)];
for k = 1:N
    states_curr = pred_state_matrix(:,k); control_curr =↩
        U(:,k);
    ARC_term1 = sin(P(n_states+3))*(states_curr(1)-P(↩
      n_states+1)) - cos(P(n_states+3))*(states_curr(2)-↩
      P(n_states+2));
```

```
        [Fzr_left,Fzr_right,Fzf_left, Fzf_right] = ↩
           vehi_safety_slope(states_curr(4),states_curr(5),↩
           states_curr(6),(control_curr(1)*0.1),(control_curr↩
           (2)*0.1),slope(1),slope(2));
        Fzr_L_term = tanh(-(Fzr_left-a_term)/b_term);↩
           Fzr_R_term = tanh(-(Fzr_right-a_term)/b_term);
        Fzf_L_term = tanh(-(Fzf_left-a_term)/b_term);↩
           Fzf_R_term = tanh(-(Fzf_right-a_term)/b_term);
        ARC_term2 = Fzr_L_term + Fzr_R_term + Fzf_L_term + ↩
           Fzf_R_term;
        Effi_term = P(n_states+n_ref_states+1)-P(n_states+↩
           n_ref_states+2);
        states_term = (states_curr(1:4)-P((n_states+1):(↩
           n_states+n_ref_states)));
%       % adding a term for hdng error:
        hdng_err_rad_term = hdng_pred_matrix(k,:);
        obj = obj + ARC_term1'*w_phi_f*ARC_term1+w_Fz*↩
           ARC_term2 + w_hdng_err*hdng_err_rad_term^2 + ↩
           states_term'*Q*states_term + Q_delta*states_curr↩
           (8)^2 + control_curr'*R*control_curr + Effi_term'*↩
           w_FE*Effi_term;
        states_next = pred_state_matrix(:,k+1);      % ↩
           define each node states with a symbolic varibale ↩
           of pred_state_matrix
        model_fn_value = model_fn(states_curr,control_curr,↩
           slope);
        states_next_euler = states_curr + (control_curr(3)*↩
           model_fn_value);
        g = [g; states_next-states_next_euler];          % ↩
           compute the constraints for each state at each ↩
           nodeend
        g = [g; -(Fzr_left-Fz_thr)];                     % ↩
           the min load on left tire should be more than 1000↩
            N
```

```
    g = [g; -(Fzr_right-Fz_thr)];                    % ↩
       the min load on left tire should be more than 1000↩
        N
    g = [g; -(Fzf_left-Fz_thr)];                    % ↩
       the min load on left tire should be more than 1000↩
        N
    g = [g; -(Fzf_right-Fz_thr)];                    % ↩
       the min load on left tire should be more than 1000↩
        N
    pred_dist = states_curr(4)*control_curr(3)*N;
    g = [g; -(pred_dist-Lidar_dist_limit)];      % this ↩
       would decide the min distance that can be made ↩
       predictions,
end
s_0 = sqrt((P(n_states+1)-P(1))^2+(P(n_states+2)-P(2))↩
   ^2);
s_f = sqrt((P(n_states+1)-states_curr(1))^2+(P(n_states↩
   +2)-states_curr(2))^2);
phi_frg = atan2(P(n_states+2)-states_curr(2),P(n_states↩
   +1)-states_curr(1));
phi_diff = atan2(sin(states_curr(3)-phi_frg),cos(↩
   states_curr(3)-phi_frg));
obj = obj + w_dist*s_f/s_0 + w_phi*phi_diff^2+w_t*↩
   control_curr(3)*N;
obs_dia = 5.0;          % in meters
for O = 1:no_of_obs_detect
    obj = obj + w_obs/(sqrt((pred_state_matrix(1,k)-P(↩
       n_states+n_ref_states+n_FE_states+(2*O-1)))^2 + (↩
       pred_state_matrix(2,k)-P(n_states+n_ref_states+↩
       n_FE_states+2*O))^2)+(rob_diam/2 + obs_dia/2));
end
c1 = -1.28e-4; c2=8.59e-3; c3=-0.2257; c4=3.0828; c5↩
   =-1.38e-4; c6=6.85e-3;c7=-0.1204;c8=-3.5589;
ax_max = c1*vx^3+c2*vx^2+c3*vx+c4;
```

```
ax_min = c5*vx^3+c6*vx^2+c7*vx+c8;
constraint_model = [ax_max;ax_min];
con_fn = Function('f',{states},{constraint_model});
for k = 1:N+1
    for O = 1:no_of_obs_detect    % for number of ↩
       obstacles, define constraint for each obstacle
        g = [g; -sqrt((pred_state_matrix(1,k)-P(n_states↩
          +n_ref_states+n_FE_states+(2*O-1)))^2 + (↩
          pred_state_matrix(2,k)-P(n_states+n_ref_states↩
          +n_FE_states+(2*O)))^2)+(rob_diam/2 + obs_dia↩
          /2)];
    end
end
OPT_variables = [reshape(pred_state_matrix,(n_states)*(N↩
   +1),1);reshape(U,n_controls*N,1)];
prob_NLP = struct('f',obj, 'x', OPT_variables, 'g', g, '↩
   p', P);
%STEP 5: define solver settings and assign it to a ↩
   object
opts = struct;
opts.ipopt.max_iter = 100;  % max iteration for the ↩
   given prob
opts.ipopt.print_level = 0; %0,3
opts.print_time = 0; %0, 1
opts.ipopt.acceptable_tol = 1e-8;
opts.ipopt.acceptable_obj_change_tol = 1e-6;    % ↩
   optimality convergence tolerance
S = nlpsol('solver','ipopt',prob_NLP, opts);
% STEP 6: define bounds on constraints and states
args = struct;
args.lbg(1:n_states) = 0;
args.ubg(1:n_states) = 0;
% next is to apply constraints for the 'N' loop (↩
   prediction loop) constraints
```

```
factor = n_states;
pred_st_con_len= zeros(n_states,N);
pred_safty_con_len = zeros(n_veh_safty_states,N);
pred_dist_con_len = zeros(n_pred_len,N);
for k = 1:N
    pred_st_con_len(:,k) = (factor+1):(factor+n_states);
    pred_safty_con_len(:,k) = ((factor+n_states)+1):(↩
       factor+(n_states+n_veh_safty_states));
    pred_dist_con_len(:,k) = (factor+(n_states+↩
       n_veh_safty_states)+1):(factor+(n_states+↩
       n_veh_safty_states+n_pred_len));
    factor = factor+(n_states+n_veh_safty_states+↩
       n_pred_len);
end
args.lbg(reshape(pred_st_con_len,1,(n_states*N))) = 0;
args.ubg(reshape(pred_st_con_len,1,(n_states*N))) = 0;
args.lbg(reshape(pred_safty_con_len,1,(↩
   n_veh_safty_states*N))) = -inf;
args.ubg(reshape(pred_safty_con_len,1,(↩
   n_veh_safty_states*N))) = -5;
args.lbg(reshape(pred_dist_con_len,1,(n_pred_len*N))) = ↩
   -1;
args.ubg(reshape(pred_dist_con_len,1,(n_pred_len*N))) = ↩
   1;
args.lbg(n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+1 : n_states*(N+1)+(n_veh_safty_states*N↩
   )+(n_pred_len*N)+no_of_obs_detect*(N+1)) = -inf;
args.ubg(n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+1 : n_states*(N+1)+(n_veh_safty_states*N↩
   )+(n_pred_len*N)+no_of_obs_detect*(N+1)) = -5.0;
args.lbw(1:n_states:n_states*(N+1),1) = -705;
args.ubw(1:n_states:n_states*(N+1),1) = 705;
args.lbw(2:n_states:n_states*(N+1),1) = -705;
args.ubw(2:n_states:n_states*(N+1),1) = 705;
args.lbw(3:n_states:n_states*(N+1),1) = -2*pi;
args.ubw(3:n_states:n_states*(N+1),1) = 2*pi;
%## new addition for providing constraint for remaining ↩
   staes
args.lbw(4:n_states:n_states*(N+1),1) = 5.0;
args.ubw(4:n_states:n_states*(N+1),1) = 29;
args.lbw(5:n_states:n_states*(N+1),1) = -inf;
args.ubw(5:n_states:n_states*(N+1),1) = inf;
args.lbw(6:n_states:n_states*(N+1),1) = -inf;
args.ubw(6:n_states:n_states*(N+1),1) = inf;
args.lbw(8:n_states:n_states*(N+1),1) = -pi/6;
args.ubw(8:n_states:n_states*(N+1),1) = pi/6;
J_max = 5; J_min = - J_max;
str_rate_max =5*pi/180; str_rate_min=-str_rate_max;
% bounds on control inputs
args.lbw(n_states*(N+1)+1:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = J_min;
args.ubw(n_states*(N+1)+1:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = J_max;
args.lbw(n_states*(N+1)+2:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = str_rate_min;
args.ubw(n_states*(N+1)+2:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = str_rate_max;
args.lbw(n_states*(N+1)+3:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = 0.05;
args.ubw(n_states*(N+1)+3:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = 50.05;
% STEP7: Let's start SIMULATION LOOP: provide initial ↩
   guess and define parameters
t0 = 0;      % initial time
x0 = [0.0; 0.0; (pi/4); 20; 0.0; 0.0;0.0;0.0];
x_ref = [299; 299.5;(pi/4); 5];
x_tar = x_ref;
obs_type = 5;
```

```
const_spd_case = 0;
if obs_type == 1
    x_loc_m = [6;110];   y_loc_m = [42;100];
    long_obs_len = [15;15];
    hor_obs_len = [15;15];
    n_indi_obs = 0; % individual obs length
    [no_of_obs,obs_pos,obs_coord] = obs_maker_adv_plots(↩
       long_obs_len,hor_obs_len,n_indi_obs,x_loc_m,↩
       y_loc_m);
    % signle bump:
    bumps_coord = [250;200;10];
    bumps_pos_xyz = bumps_coord;
elseif obs_type == 2
    load obs_values_for_mvobs_ba5_type1_path_1
    bumps_coord = [250;200;10];
    bumps_pos_xyz = bumps_coord;
elseif obs_type == 3
    x_loc_m = [35;60];   y_loc_m = [80;125];
    long_obs_len = [15;60];
    hor_obs_len = [15;25];
    n_indi_obs = 0; % individual obs length
    [no_of_obs,obs_pos,obs_coord] = obs_maker_adv_plots(↩
       long_obs_len,hor_obs_len,n_indi_obs,x_loc_m,↩
       y_loc_m);
    % signle bump:
    bumps_coord = [250;200;10];
    bumps_pos_xyz = bumps_coord;
elseif obs_type == 4
    x_loc_m = [50;140];   y_loc_m = [100;160];
    long_obs_len = [15;15];
    hor_obs_len = [10;10];
    n_indi_obs = 0; % individual obs length
    [no_of_obs,obs_pos,obs_coord] = obs_maker_adv_plots(↩
       long_obs_len,hor_obs_len,n_indi_obs,x_loc_m,↩
       y_loc_m);
    hold on
%      load contour_map_path4
    load contour_map_second_paper
    [c1,h1]=contourf(x_terr_map,y_terr_map,z_terr_map);
    clabel(c1,h1);
    colorbar
    hold on
elseif obs_type == 5
    x_loc_m = [6;80;140];    y_loc_m = [42;100;160];
    long_obs_len = [15;15];
    hor_obs_len = [10;10];
    n_indi_obs = 30; % individual obs length
    [no_of_obs,obs_pos,obs_coord] = obs_maker_adv_plots(↩
       long_obs_len,hor_obs_len,n_indi_obs,x_loc_m,↩
       y_loc_m);
    hold on
    % code for creating bumps:
    load contour_map_second_paper
    [c1,h1]=contourf(x_terr_map,y_terr_map,z_terr_map);
    clabel(c1,h1);
    colorbar
elseif obs_type == 6
    x_loc_m = [50;140];    y_loc_m = [100;160];
    long_obs_len = [15;15];
    hor_obs_len = [10;10];
    n_indi_obs = 0; % individual obs length
    [no_of_obs,obs_pos,obs_coord] = obs_maker_adv_plots(↩
       long_obs_len,hor_obs_len,n_indi_obs,x_loc_m,↩
       y_loc_m);
    hold on
    % code for creating bumps:
    load contour_map_second_paper
    [c1,h1]=contourf(x_terr_map,y_terr_map,z_terr_map);
    clabel(c1,h1);
    colorbar
end
obs_his(:,1) = obs_pos;
detected_obs_pos = ones(n_obs_detected,1)*1000;
detected_obs_pos_his(:,1) = detected_obs_pos;
xx(:,1) = x0;
t(1) = t0;
hdng_err_init = 0.05;
% for storing vehicle safety constraints stuff
veh_sfty_N(:,1) = [3;3;3;3]; % arbitrary values above 1 ↩
   KN
u0_int = [0.025;0.0015;0.06];
ux(:,1) = u0_int;
u0 = repmat(u0_int,1,N)';
% Engine/Motor Efficiency values
effi_Tar = 94;  [rpm,T_eng,effi_curr] = LVD_model(x0(4),↩
   u0_int(1));
vehi_oper_pts(:,1) = [effi_Tar;rpm;T_eng;effi_curr]; ↩
        % save the data in eff term
pred_state_mat_init = repmat(x0,1,N+1)';
hdng_err_mat_init = repmat(hdng_err_init,1,N+1)';
slope_rad = [0.5;0.5];
con_fn_value =con_fn(x0);     % use current states and ↩
   control inputs and calculate ODE fn value
args.lbw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(2));   % lower bound on acce, (Use eqn s ↩
   for it)
args.ubw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(1));
sim_time = 100;        % was 40max simulation for the ↩
   given target
```

```
% start MPC from here:
mpciter = 0;
% store the prediction horizon results
xx1 = [];        % for storing vehicle states along the ↩
   horizon
u_cl = [];       % for storing control inputs along the ↩
   horizon
veh_sfty_N1 = [];     % for storing the vehicle safety ↩
   constraints along the horizon
main_loop = tic;
obs_info = []; obs_info_out = [];
up_limit = hdng_mapping(x_tar(3) + pi/30);lp_limit = ↩
   hdng_mapping(x_tar(3) - pi/30);
while(norm((x0(1:2)-x_ref(1:2)),2) > 159e-1 && mpciter <↩
    sim_time/0.1)
    %STEP 7: define initial values and refernce values
    args.p = [x0;x_ref;effi_Tar;effi_curr;↩
       detected_obs_pos;slope_rad;hdng_err_mat_init];    %↩
        set values of 'p' with initial and reference ↩
       values. if we need trajectory tracking, update the↩
        'w_ref' values for each iteration
    args.w_init = [reshape(pred_state_mat_init',n_states↩
       *(N+1),1);reshape(u0',n_controls*N,1)];    % ↩
       initial value of the optimization varibales
    % STEP 8: assign the values to Casadi object 'S' and↩
        prodcue the results
    sol = S('x0',args.w_init, 'lbx', args.lbw, 'ubx', ↩
       args.ubw,...
        'lbg',args.lbg, 'ubg', args.ubg, 'p', args.p);
    u = reshape(full(sol.x(n_states*(N+1)+1:end))',↩
       n_controls,N)';
    xx1(:,1:n_states,mpciter+1) = reshape(full(sol.x(1:↩
       n_states*(N+1)))',n_states,N+1)';
    u_cl(:,1:n_controls,mpciter+1) = u;
```

```
% store the vehi safety vertical load values from ↩
   rear left and rear tires for making plots,
veh_sfty = (-1*(reshape(full(sol.g(reshape(↩
   pred_safty_con_len,1,(n_veh_safty_states*N))))',↩
   n_veh_safty_states,N)')+Fz_thr)/1000;
veh_sfty_N1(:,1:n_veh_safty_states,mpciter+1) = ↩
   veh_sfty;
obs_dist_constr(:,1,mpciter+1) = (-1*full(sol.g(↩
   n_states*(N+1)+(n_veh_safty_states*N)+(n_pred_len*↩
   N)+1 : n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+no_of_obs_detect*(N+1))));
% update the initial states
pred_state_matrix = reshape(full(sol.x(1:n_states*(N↩
   +1)))',n_states,N+1);
t(mpciter+1) = t0;
[t0, x0, u0] = shift( dt, t0, x0, u, slope_rad, ↩
   model_fn);
% add effieciny_curr term here:
[rpm,T_eng,effi_curr] = LVD_model(x0(4),u(1,1));
con_fn_value =con_fn(x0);     % use current states ↩
   and control inputs and calculate ODE fn value
% bounds on control inputs
args.lbw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(2));   % lower bound on acce, (Use ↩
   eqn s for it)
args.ubw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(1));
x0(3) = hdng_mapping(x0(3));
[tar_hdng,x_ref(4),obs_info,slope_deg] = ↩
   box_obs_slope_contour_search(x0(1:3),x_tar(1:3),↩
   obs_coord,x_terr_map,y_terr_map,z_terr_map);
if const_spd_case
     x_ref(4) = 20;  % make simulation with costant ↩
        speed of 20 m/sec
```

```
end
if (length(obs_info)) > 0
    obs_info_out = [obs_info_out',obs_info];
    obs_info_out_coord = reshape(obs_info_out,2,↩
       length(obs_info_out)/2);
    [UniXY,Index]=unique(obs_info_out_coord','rows')↩
       ;
    obs_info_out = reshape(UniXY',(size(UniXY,1)*2)↩
       ,1);
    k_del = [];
    for i = 1:length(obs_info_out)/2
        dist_m = sqrt((obs_info_out(2*i-1) - x0(1))↩
           ^2 + (obs_info_out(2*i) - x0(2))^2);
        if dist_m > 125 && i > 2
            k_del = [k_del, (2*i-1),(2*i)];
        end
    end
    obs_info_out(k_del,:) = []; % remove the obs, ↩
       that are far from vehicle current position,
    if (length(obs_info_out)) <= n_obs_detected
        detected_obs_pos(1:length(obs_info_out),1) =↩
            obs_info_out;
    else
        detected_obs_pos = obs_info_out(1:↩
           n_obs_detected,1);
    end
end
x_ref(3) =  tar_hdng;
slope_rad(1) = slope_deg(1)*pi/180;
slope_rad(2) = slope_deg(2)*pi/180;
xx(:,mpciter+2) = x0;   % take the history of sates ↩
   and stores it into this parameter
ux(:,mpciter+2) = u(1,:);   % take the history of ↩
   control inputs and store it in this parameter
```

```
    veh_sfty_N(:,mpciter+2) = veh_sfty(1,:);
    obs_his(:,mpciter+2) = obs_pos; % take the history ↩
       of obstacle information
    detected_obs_pos_his(:,mpciter+2) = detected_obs_pos↩
       ; % for plotting the obstacles detected along the ↩
       way
    vehi_oper_pts(:,mpciter+2) = [effi_Tar;rpm;T_eng;↩
       effi_curr];        % for making efficiency plots
    pred_state_mat_init = reshape(full(sol.x(1:n_states↩
       *(N+1)))',n_states,N+1)';  % this is solution ↩
       trajectory, i.e, optimal states at each node
    pred_state_mat_init = [pred_state_mat_init(2:end,:);↩
       pred_state_mat_init(end,:)];
    hdng_err_mat_init = hdn_err_fn(pred_state_mat_init,↩
       x_ref(3));
    mpciter
    x0(1),x0(2)
    mpciter = mpciter+1;
end
main_loop_time = toc(main_loop);
ss_error = norm((x0(1:3)-x_ref(1:3)),2)
average_mpc_time = main_loop_time/(mpciter+1)
```

B.2 Moving Obstacle avoidance code

```
%% step 1: define decision variables
clear all
close all
clc
format compact
 % to add casadi path to the program
addpath('G:\My Drive\PhD_work\AV2_Project\N-MPC_stuff\↩
   casadi-windows-matlabR2016a-v3.4.5');
```

```
import casadi.*
% STEP 1: define constants and design varibales
dt = 0.05;
N = 50;
% vehicle specs:
mass = 2689;     % in kgs
lf = 1.58; lr = 1.72; % front,rear axle to C.G. location
Iz = 4110;       % in kg-m^2
rob_diam = 2.5;
L = lf+lr;        % wheel base
delta_f_max = pi/4; delta_f_min = -delta_f_max;
%## define control inputs
jerk = SX.sym('Jk');
str_rate = SX.sym('str_rate');
T = SX.sym('Tp');      % prediction horizon as design ↩
   variable
%## define states
Pos_x = SX.sym('s1');
Pos_y = SX.sym('s2');
theta = SX.sym('theta');
vx = SX.sym('vx');
vy = SX.sym('vy');          % define y_dot
wz = SX.sym('wz');
ax = SX.sym('acce');  % define varibale long velo, vx in↩
    m/sec
delta_f = SX.sym('del_f');    % steering angle
% let's add 6 more states for 3 moving obstacles and and↩
    try to include them for prediction analysis,
obs_states=2;
n_mvobs = 3; n_mvobs_states =  n_mvobs*obs_states;
mvobs1 = SX.sym('mv1');mvobs2 = SX.sym('mv2');mvobs3 = ↩
   SX.sym('mv3');mvobs4 = SX.sym('mv4');mvobs5 = SX.sym('↩
   mv5');mvobs6 = SX.sym('mv6');
```

```
mvobs_states = SX.sym('mvobs',(n_mvobs_states),1); % ↩
   this is in order of [mvobs_x1;mvobs_y1;mvobs_x2;↩
   mvobs_y2;mvobs_x3;mvobs_y3],
% FE_states in symbolic fashion:
FE_effi_Tar = SX.sym('Effi_Tar');
FE_effi_req = SX.sym('Effi_req');
states = [Pos_x;Pos_y;theta;vx;vy;wz;ax;delta_f;mvobs1;↩
   mvobs2;mvobs3;mvobs4;mvobs5;mvobs6];
n_states = length(states);
n_ref_states = 4;                % x_tar,y_tar,theta_tar,↩
   velo_tar (adding velo too)
controls = [jerk;str_rate;T];         % these are control↩
    inputs
n_controls = length(controls);
FE_states = [FE_effi_Tar;FE_effi_req];
n_FE_states = length(FE_states);
% define length of vehicle safety constrints,
n_veh_safty_states = 4;     % vehicle safety is applied ↩
   on one for left-rear tire and second on right-rear ↩
   tire.
no_of_obs_detect = 56; % was 76
n_obs_detected = no_of_obs_detect*obs_states;
% apply moving constraint for obstacle avoidance
n_min_dist = 1; % all vehicle prediction states should ↩
   be min 2.5 m from the nearest obstacle
n_slope = 2;    % this state is to account for road ↩
   slope and thus reduce the speed of the vehicle or ↩
   avoid the slope
n_mvobs_velo=3;
n_mvobs_theta=3; % these values will be updated from ↩
   mv_obs_search fn,
n_hdng_err = (N+1); % for making the hdng error values
U = SX.sym('U',n_controls, N);               % symbolic ↩
   control inputs defined at each prediction node
```

```
P = SX.sym('P',n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope+n_mvobs_velo+n_mvobs_theta+↩
   n_hdng_err);      % here, we would consider only Pos_x↩
   ,Pos_y,theta can be target values to be minimised
pred_state_matrix = SX.sym('X_pred',n_states,(N+1));    ↩
   % this is a symbolic 3x4 state matrix at each ↩
   prediction node, which is 1 unit extra in comparison ↩
   to input_matrix.
xdot = states(4)*cos(states(3)) - (states(5)+lf*states↩
   (6))*sin(states(3)); % added (V+Lf*wz) on Jan 16,2020
ydot = states(4)*sin(states(3)) + (states(5)+lf*states↩
   (6))*cos(states(3));
thetadot = states(6); % + states(4)/L*tan(states(8)), ↩
   added by ABD for including the effect of steering ↩
   commands on vehicle yaw,
xdotdot = states(7); %;
slope= [P(n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+1);P(n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope)];
[Fyf,Fyr] = Tire_model(states(4),states(5),states(6),↩
   states(8),states(7),slope(1));
ydotdot = (Fyf+Fyr)/mass - states(4)*states(6);
thetadotdot = (Fyf*lf - Fyr*lr)/Iz;
acce_dot = controls(1);
delta_dot = controls(2);
obs_vel = P(n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope+1:n_states+n_ref_states+↩
   n_FE_states+n_obs_detected+n_slope+n_mvobs_velo);
obs_theta = P(n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope+n_mvobs_velo+1:n_states+↩
   n_ref_states+n_FE_states+n_obs_detected+n_slope+↩
   n_mvobs_velo+n_mvobs_theta);
% add Euler logic for moving obsatcles
```

```
mvobs_xdot1 = obs_vel(1)*cos(obs_theta(1)); % this is ↩
   simply, v_obs1*cos(obs_theta1)
mvobs_ydot1 = obs_vel(1)*sin(obs_theta(1)); % this is ↩
   simply, v_obs1*cos(obs_theta1)
mvobs_xdot2 = obs_vel(2)*cos(obs_theta(2)); % this is ↩
   simply, v_obs1*cos(obs_theta1)
mvobs_ydot2 = obs_vel(2)*sin(obs_theta(2)); % this is ↩
   simply, v_obs1*cos(obs_theta1)
mvobs_xdot3 = obs_vel(3)*cos(obs_theta(3)); % this is ↩
   simply, v_obs1*cos(obs_theta1)
mvobs_ydot3 = obs_vel(3)*sin(obs_theta(3)); % this is ↩
   simply, v_obs1*cos(obs_theta1)
% add hdng error through P-parameter
hdng_pred_matrix = P(n_states+n_ref_states+n_FE_states+↩
   n_obs_detected+n_slope+n_mvobs_velo+n_mvobs_theta+1:↩
   n_states+n_ref_states+n_FE_states+n_obs_detected+↩
   n_slope+n_mvobs_velo+n_mvobs_theta+n_hdng_err);
sys_model = [xdot;ydot;thetadot;xdotdot;ydotdot;↩
   thetadotdot;acce_dot;delta_dot;mvobs_xdot1;mvobs_ydot1↩
   ;mvobs_xdot2;mvobs_ydot2;mvobs_xdot3;mvobs_ydot3];
model_fn = Function('f',{states,controls,slope,obs_vel,↩
   obs_theta},{sys_model});
% STEP 2: formulate Objective function:
obj = 0;    % initialize with objective fn
g = [];     % constraint vector
% integral gains:
Q = zeros(4,4);
Q(1,1) = 0.05; Q(2,2) = 0.05;  Q(3,3) = 0.0; Q(4,4)= 50;
w_hdng_err = 60; Q_delta = 0.005;
R = zeros(3,3); R(1,1) = 0.05; R(2,2) = 0.5;  R(3,3) = ↩
   0.00;
w_FE = 0.0; w_phi_f = 5e-2;w_Fz = 0.005;
% terminal cost weights
w_dist = 0.5;w_phi = 15; w_obs = 0.005; w_t = 0.001;
```

```
% not yet used weight terms:
w_delta = 0.1; w_str = 1; w_j = 0.01; w_cf = 1;
% vehicle safety constraints terms:
Fz_thr = 1000; % Threshold vertical load for vehicle ↩
   safety constraint, in N
Fz_off = 300;   % in N
a_term = Fz_thr + 3*Fz_off;
b_term = Fz_off;
Lidar_dist_limit = 75;      % was 125, 75, can varied ↩
   based on the lidar distance availability,
% define parameter for making equal prediction length,
n_pred_len = 1;
states_curr = pred_state_matrix(:,1);
g = [g; states_curr-P(1:n_states)];
for k = 1:N
    states_curr = pred_state_matrix(:,k); control_curr =↩
        U(:,k);
    ARC_term1 = sin(P(n_states+3))*(states_curr(1)-P(↩
       n_states+1)) - cos(P(n_states+3))*(states_curr(2)-↩
       P(n_states+2));
    [Fzr_left,Fzr_right,Fzf_left, Fzf_right] = ↩
       vehi_safety_slope(states_curr(4),states_curr(5),↩
       states_curr(6),(control_curr(1)*0.1),(control_curr↩
       (2)*0.1),slope(1),slope(2));
    Fzr_L_term = tanh(-(Fzr_left-a_term)/b_term);↩
       Fzr_R_term = tanh(-(Fzr_right-a_term)/b_term);
    Fzf_L_term = tanh(-(Fzf_left-a_term)/b_term);↩
       Fzf_R_term = tanh(-(Fzf_right-a_term)/b_term);
    ARC_term2 = Fzr_L_term + Fzr_R_term + Fzf_L_term + ↩
       Fzf_R_term;
    Effi_term = P(n_states+n_ref_states+1)-P(n_states+↩
       n_ref_states+2);
    states_term = (states_curr(1:4)-P((n_states+1):(↩
       n_states+n_ref_states)));
```

```
hdng_err_rad_term = hdng_pred_matrix(k,:);
obj = obj + ARC_term1'*w_phi_f*ARC_term1+w_Fz*↩
   ARC_term2 + hdng_err_rad_term*w_hdng_err*↩
   hdng_err_rad_term + states_term'*Q*states_term + ↩
   0.005*states_curr(8)^2 + control_curr'*R*↩
   control_curr + Effi_term'*w_FE*Effi_term;
states_next = pred_state_matrix(:,k+1);       % ↩
   define each node states with a symbolic varibale ↩
   of pred_state_matrix
model_fn_value = model_fn(states_curr,control_curr,↩
   slope,obs_vel,obs_theta);      % use current states↩
    and control inputs and calculate ODE fn value
states_next_euler = states_curr + (control_curr(3)*↩
   model_fn_value);           % use the above ODE value↩
   , current states,current control inputs and DT to ↩
   predcit the next time step state values. this is ↩
   called numerical integration.
g = [g; states_next-states_next_euler];          % ↩
   let's make chnage here..we don't need any equality↩
    constraint for mvobs, compute the constraints for↩
    each state at each nodeend
g = [g; -(Fzr_left-Fz_thr)];                     % ↩
   the min load on left tire should be more than 1000↩
    N
g = [g; -(Fzr_right-Fz_thr)];                     % ↩
   the min load on left tire should be more than 1000↩
    N
g = [g; -(Fzf_left-Fz_thr)];                     % ↩
   the min load on left tire should be more than 1000↩
    N
g = [g; -(Fzf_right-Fz_thr)];                     % ↩
   the min load on left tire should be more than 1000↩
    N
    % add final constraint for making the equal distance↩
        predictions with the equal distance predicitons,
    pred_dist = states_curr(4)*control_curr(3)*N;
    g = [g; -(pred_dist-Lidar_dist_limit)];      % this ↩
       would decide the min distance that can be made ↩
       predictions,
end
% Add ARC terminal cost cost function:
s_0 = sqrt((P(n_states+1)-P(1))^2+(P(n_states+2)-P(2))↩
   ^2);
s_f = sqrt((P(n_states+1)-states_curr(1))^2+(P(n_states↩
   +2)-states_curr(2))^2);
phi_frg = atan2(P(n_states+2)-states_curr(2),P(n_states↩
   +1)-states_curr(1));
phi_diff = atan2(sin(states_curr(3)-phi_frg),cos(↩
   states_curr(3)-phi_frg));
obj = obj + w_dist*s_f/s_0 + w_phi*phi_diff^2+w_t*↩
   control_curr(3)*N;
% adding obstacle avoidance soft constraint in cost ↩
   function:
obs_dia = 5.0;           % in meters
for O = 1:no_of_obs_detect   % for number of obstacles, ↩
   define constraint for each obstacle
    obj = obj + w_obs/(sqrt((pred_state_matrix(1,k)-P(↩
       n_states+n_ref_states+n_FE_states+(2*O-1)))^2 + (↩
       pred_state_matrix(2,k)-P(n_states+n_ref_states+↩
       n_FE_states+2*O))^2)+(rob_diam/2 + obs_dia/2));
end
% limits on prediction horizon Time:
N_max =  s_0/(N*5); N_min = s_0/(N*29); % provide max, ↩
   min values for prediction horizon
c1 = -1.28e-4; c2=8.59e-3; c3=-0.2257; c4=3.0828; c5↩
   =-1.38e-4; c6=6.85e-3;c7=-0.1204;c8=-3.5589;
ax_max = c1*vx^3+c2*vx^2+c3*vx+c4;
```

```
ax_min = c5*vx^3+c6*vx^2+c7*vx+c8;
constraint_model = [ax_max;ax_min];
con_fn = Function('f',{states},{constraint_model});
for k = 1:N+1
    for O = 1:no_of_obs_detect   % for number of ↩
       obstacles, define constraint for each obstacle
        g = [g; -sqrt((pred_state_matrix(1,k)-P(n_states↩
           +n_ref_states+n_FE_states+(2*O-1)))^2 + (↩
           pred_state_matrix(2,k)-P(n_states+n_ref_states↩
           +n_FE_states+(2*O)))^2)+(rob_diam/2 + obs_dia↩
           /2)];
    end
end
% let's add moving obstacle avoidance constraint logic,
mv_fact = 1;
for k = 1:N+1
    for O = 1:n_mvobs   % for number of obstacles, ↩
       define constraint for each obstacle
        for h = 1:(N/mv_fact)   % for each prediction ↩
           step value of moving obstacle,
            g = [g; -sqrt((pred_state_matrix(1,k)-↩
               pred_state_matrix((6+(2*O-1)),h))^2 + (↩
               pred_state_matrix(2,k)-pred_state_matrix↩
               ((6+(2*O)),h))^2)+(rob_diam/2 + obs_dia/2)↩
               ];
        end
    end
end
OPT_variables = [reshape(pred_state_matrix,n_states*(N↩
   +1),1);reshape(U,n_controls*N,1)];
prob_NLP = struct('f',obj, 'x', OPT_variables, 'g', g, '↩
   p', P);
%STEP 5: define solver settings and assign it to a ↩
   object
```

```
opts = struct;
opts.ipopt.max_iter = 100;  % max iteration for the ↩
   given prob
opts.ipopt.print_level = 0;
opts.print_time = 0; %0, 1
opts.ipopt.acceptable_tol = 1e-8;
opts.ipopt.acceptable_obj_change_tol = 1e-6;
S = nlpsol('solver','ipopt',prob_NLP, opts);
% STEP 6: define bounds on constraints and states
args = struct;
args.lbg(1:n_states) = 0;    %-1e-20 % equality ↩
   constraint i.e, this is a constraint at each node for ↩
   each state varibale saying that,
args.ubg(1:n_states) = 0;     % 1e-20 % equality ↩
   constraints, similarly for the upper bound too
factor = n_states;
pred_st_con_len= zeros(n_states,N);             % size of↩
    vehi state constraints
pred_safty_con_len = zeros(n_veh_safty_states,N);  % ↩
   size of vehicle safety constraints
pred_dist_con_len = zeros(n_pred_len,N);  % for making ↩
   equal length predictions for fixed distance
for k = 1:N
    pred_st_con_len(:,k) = (factor+1):(factor+n_states);↩
            % take out every six constraints and keep it ↩
       in x_len
    pred_safty_con_len(:,k) = ((factor+n_states)+1):(↩
       factor+(n_states+n_veh_safty_states)); % take out ↩
       7th and 8th constraints and store it in one ↩
       parameters
    pred_dist_con_len(:,k) = (factor+(n_states+↩
       n_veh_safty_states)+1):(factor+(n_states+↩
       n_veh_safty_states+n_pred_len)); % take out 7th ↩
       and 8th constraints and store it in one parameter
    factor = factor+(n_states+n_veh_safty_states+↩
       n_pred_len);
end
args.lbg(reshape(pred_st_con_len,1,(n_states*N))) = 0; ↩
       % this is our final vector to be applied ↩
   constraints for states
args.ubg(reshape(pred_st_con_len,1,(n_states*N))) = 0;
args.lbg(reshape(pred_safty_con_len,1,(↩
   n_veh_safty_states*N))) = -inf;     % this is for ↩
   safety constraints
args.ubg(reshape(pred_safty_con_len,1,(↩
   n_veh_safty_states*N))) = -5;
args.lbg(reshape(pred_dist_con_len,1,(n_pred_len*N))) = ↩
   -1;     % this is for safety constraints
args.ubg(reshape(pred_dist_con_len,1,(n_pred_len*N))) = ↩
   1;
args.lbg(n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+1 : n_states*(N+1)+(n_veh_safty_states*N↩
   )+(n_pred_len*N)+no_of_obs_detect*(N+1)) = -inf;    %↩
   inequality constraints for obstacle avoidance
args.ubg(n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+1 : n_states*(N+1)+(n_veh_safty_states*N↩
   )+(n_pred_len*N)+no_of_obs_detect*(N+1)) = -1.0;    % ↩
   upper bound should go from 0 to -inf side (boz, cal ↩
   dist is -ve, see obs constraint logic above)
% let's add mv obs constraint as same as above
args.lbg(n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+no_of_obs_detect*(N+1)+1: n_states*(N+1)↩
   +(n_veh_safty_states*N)+(n_pred_len*N)+↩
   no_of_obs_detect*(N+1)+n_mvobs*(N+1)*(N/mv_fact)) = -↩
   inf;    %inequality constraints for obstacle avoidance
```

```
args.ubg(n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+no_of_obs_detect*(N+1)+1: n_states*(N+1)↩
   +(n_veh_safty_states*N)+(n_pred_len*N)+↩
   no_of_obs_detect*(N+1)+n_mvobs*(N+1)*(N/mv_fact)) = ↩
   -0.5;
% constraints on OPT_varibales:
args.lbw(1:n_states:n_states*(N+1),1) = -705;
args.ubw(1:n_states:n_states*(N+1),1) = 705;
args.lbw(2:n_states:n_states*(N+1),1) = -705;
args.ubw(2:n_states:n_states*(N+1),1) = 705;
args.lbw(3:n_states:n_states*(N+1),1) = -2*pi;
args.ubw(3:n_states:n_states*(N+1),1) = 2*pi;
%## new addition for providing constraint for remaining ↩
   staes
args.lbw(4:n_states:n_states*(N+1),1) = 5.0;
args.ubw(4:n_states:n_states*(N+1),1) = 29;
args.lbw(5:n_states:n_states*(N+1),1) = -inf;
args.ubw(5:n_states:n_states*(N+1),1) = inf;
args.lbw(6:n_states:n_states*(N+1),1) = -inf;
args.ubw(6:n_states:n_states*(N+1),1) = inf;
args.lbw(8:n_states:n_states*(N+1),1) = -pi/6;
args.ubw(8:n_states:n_states*(N+1),1) = pi/6;
% add constraints for mv obs
args.lbw(9:n_states:n_states*(N+1),1) = -10100;
args.ubw(9:n_states:n_states*(N+1),1) = 10100;
args.lbw(10:n_states:n_states*(N+1),1) = -10100;
args.ubw(10:n_states:n_states*(N+1),1) = 10100;
args.lbw(11:n_states:n_states*(N+1),1) = -10100;
args.ubw(11:n_states:n_states*(N+1),1) = 10100;
args.lbw(12:n_states:n_states*(N+1),1) = -10100;
args.ubw(12:n_states:n_states*(N+1),1) = 10100;
args.lbw(13:n_states:n_states*(N+1),1) = -10100;
args.ubw(13:n_states:n_states*(N+1),1) = 10100;
args.lbw(14:n_states:n_states*(N+1),1) = -10100;
args.ubw(14:n_states:n_states*(N+1),1) = 10100;
J_max = 5; J_min = - J_max; Tp_max = 50.05; Tp_min = ↩
   0.05;
str_rate_max =5*pi/180; str_rate_min=-str_rate_max;
% bounds on control inputs
args.lbw(n_states*(N+1)+1:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = J_min;
args.ubw(n_states*(N+1)+1:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = J_max;
args.lbw(n_states*(N+1)+2:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = str_rate_min;
args.ubw(n_states*(N+1)+2:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = str_rate_max;
args.lbw(n_states*(N+1)+3:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = Tp_min;
args.ubw(n_states*(N+1)+3:n_controls:n_states*(N+1)+↩
   n_controls*N,1) = Tp_max;
t0 = 0;     % initial time
x0 = [0.005; 0.005; (pi/4); 20; 0.0; 0.0; 0.0; 0.0; ↩
   1000; 1000; 1000; 1000; 1000; 1000];
x_ref = [299; 299.5;(pi/4); 5];
x_tar = x_ref;
% this is only for mvobs state update fn alone: (not for↩
    st. obs)
obs_velo_update=zeros(3,1); obs_theta_update=zeros(3,1);
obs_type = 1;
% obs_type = j_k;
const_spd_case = 0; % if you would like to see const ↩
   speed case results, activate this as 1.,
if obs_type == 1
    x_loc_m = [200;30;100;50;150];   y_loc_m = ↩
       [150;200;250;100;160];
    long_obs_len = [1;1;1;15;15];
    hor_obs_len = [1;1;1;10;10];
    n_indi_obs = 0;
    [no_of_obs,obs_pos,obs_coord,obs_velo_ms,↩
       obs_theta_rad,mvobs_ind] = obs_maker_adv1_plots(↩
       long_obs_len,hor_obs_len,n_indi_obs,x_loc_m,↩
       y_loc_m);
    hold on
    % code for creating bumps:
    load contour_map_second_paper
    [c1,h1]=contourf(x_terr_map,y_terr_map,z_terr_map);
    clabel(c1,h1);
    colorbar
    hold off
elseif obs_type == 2
    x_loc_m = [200;30;100;6;50;140];    y_loc_m = ↩
       [150;200;250;42;100;160];
    long_obs_len = [1;1;1;15;15];
    hor_obs_len = [1;1;1;10;10];
    n_indi_obs = 30;
    [no_of_obs,obs_pos,obs_coord,obs_velo_ms,↩
       obs_theta_rad,mvobs_ind] = obs_maker_adv1_plots(↩
       long_obs_len,hor_obs_len,n_indi_obs,x_loc_m,↩
       y_loc_m);
    hold on
    % code for creating bumps:
    load contour_map_second_paper
    [c1,h1]=contourf(x_terr_map,y_terr_map,z_terr_map);
    clabel(c1,h1);
    colorbar
    hold off
end
obs_his(:,1) = obs_pos;
detected_obs_pos = ones(n_obs_detected,1)*1000;
detected_obs_pos_his(:,1) = detected_obs_pos;
xx(:,1) = x0;
```

```
t(1) = t0;
hdng_err_init = 0.05;   % for hdng error term
% for storing vehicle safety constraints stuff
veh_sfty_N(:,1) = [3;3;3;3]; % arbitrary values above 1 ↩
   KN
u0_int = [0.025;0.0015;0.06];
ux(:,1) = u0_int;
u0 = repmat(u0_int,1,N)';
% Engine/Motor Efficiency values
effi_Tar = 94;  [rpm,T_eng,effi_curr] = LVD_model(x0(4),↩
   u0_int(1));
vehi_oper_pts(:,1) = [effi_Tar;rpm;T_eng;effi_curr];
pred_state_mat_init = repmat(x0,1,N+1)';
hdng_err_mat_init = repmat(hdng_err_init,1,N+1)';
slope_rad = [0.5;0.5];
con_fn_value =con_fn(x0);
% bounds on control inputs
args.lbw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(2));   % lower bound on acce, (Use eqn s ↩
   for it)
args.ubw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(1));
sim_time = 100;        % was 40max simulation for the ↩
   given target
% start MPC from here:
mpciter = 0;
% store the prediction horizon results
xx1 = [];        % for storing vehicle states along the ↩
   horizon
u_cl = [];       % for storing control inputs along the ↩
   horizon
veh_sfty_N1 = [];     % for storing the vehicle safety ↩
   constraints along the horizon
main_loop = tic;
```

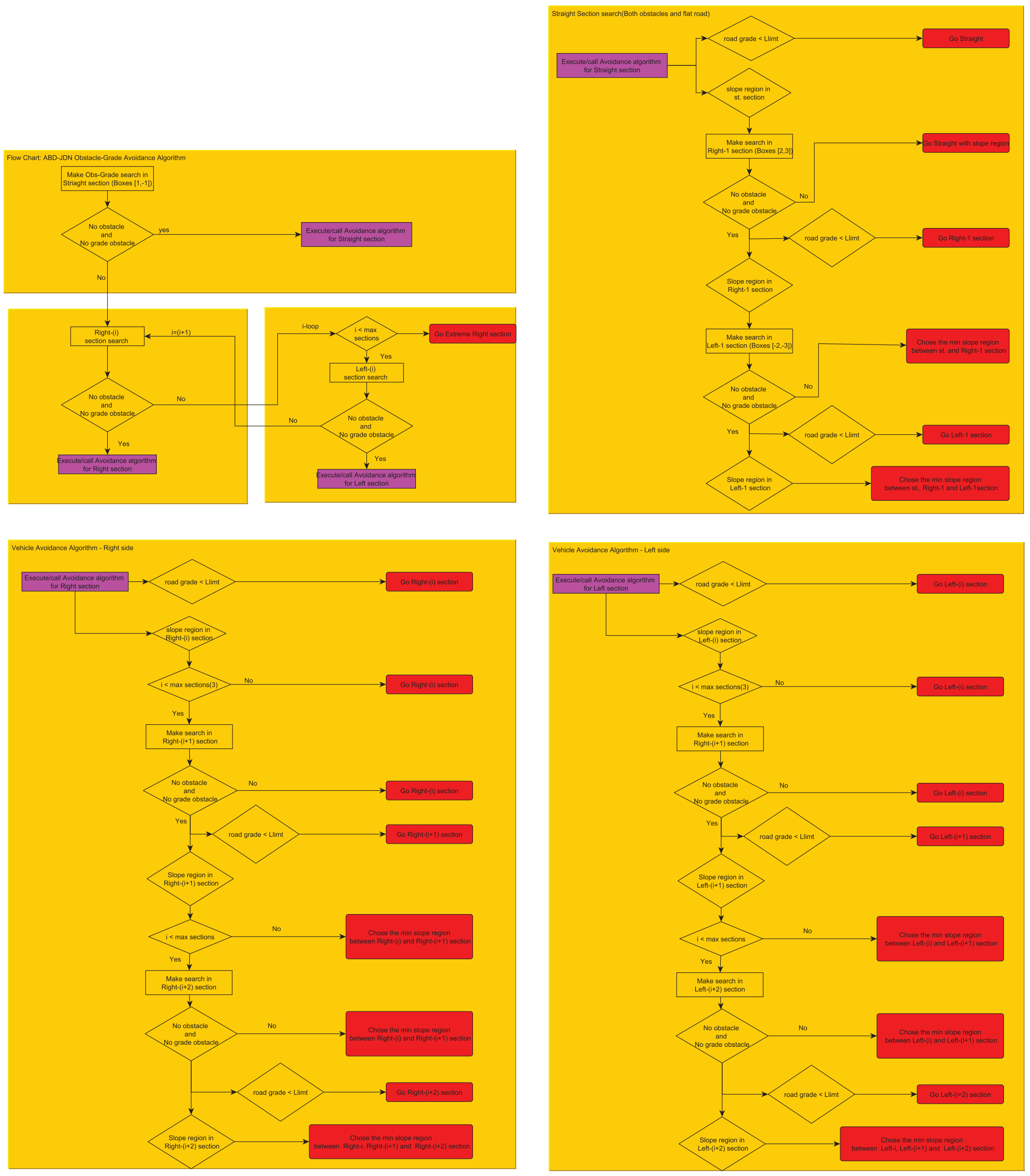
```
obs_info = []; obs_info_out = [];
mv_obs_velo_ms = [0.0001;0.0001;0.0001]; ↩
   mv_obs_theta_rad = [0.05;0.05;0.05];
detected_mov_obs_pos_his(:,1) = ones(n_mvobs*obs_states↩
   ,1)*1000;
detected_mov_obs_velo_his(:,1) = ones(n_mvobs,1)*0.005;
detected_mov_obs_theta_his(:,1) = ones(n_mvobs,1)*0.005;
% store the 'obs_info' for solving stationary obstacle ↩
   storing issue,,
obs_info_his(:,1) = zeros(n_obs_detected,1);
up_limit = hdng_mapping(x_tar(3) + pi/30);lp_limit = ↩
   hdng_mapping(x_tar(3) - pi/30);
% dist[];norm((x0-x_ref),2) > 1e-2 &&
while(norm((x0(1:2)-x_ref(1:2)),2) > 159e-1 && mpciter <↩
    800)
    %STEP 7: define initial values and refernce values
    args.p = [x0;x_ref;effi_Tar;effi_curr;↩
       detected_obs_pos;slope_rad;mv_obs_velo_ms;↩
       mv_obs_theta_rad;hdng_err_mat_init];
    args.w_init = [reshape(pred_state_mat_init',n_states↩
       *(N+1),1);reshape(u0',n_controls*N,1)];    % ↩
       initial value of the optimization varibales
    % STEP 8: assign the values to Casadi object 'S' and↩
        prodcue the results
    sol = S('x0',args.w_init, 'lbx', args.lbw, 'ubx', ↩
       args.ubw,...
        'lbg',args.lbg, 'ubg', args.ubg, 'p', args.p);
    u = reshape(full(sol.x(n_states*(N+1)+1:end))',↩
       n_controls,N)';
    xx1(:,1:n_states,mpciter+1) = reshape(full(sol.x(1:↩
       n_states*(N+1)))',n_states,N+1)';  % take 1 to 3*(↩
       N+1) terms and convert it into [3x(N+1)] matrix
    u_cl(:,1:n_controls,mpciter+1) = u;
```

```
 % store the vehi safety vertical load values from ↩
    rear left and rear tires for making plots,
veh_sfty = (-1*(reshape(full(sol.g(reshape(↩
   pred_safty_con_len,1,(n_veh_safty_states*N))))',↩
   n_veh_safty_states,N)')+Fz_thr)/1000;
veh_sfty_N1(:,1:n_veh_safty_states,mpciter+1) = ↩
   veh_sfty;
obs_dist_constr(:,mpciter+1) = (-1*full(sol.g(↩
   n_states*(N+1)+(n_veh_safty_states*N)+(n_pred_len*↩
   N)+1 : n_states*(N+1)+(n_veh_safty_states*N)+(↩
   n_pred_len*N)+no_of_obs_detect*(N+1))));
mv_obs_dist_constr(:,mpciter+1) = (-1*full(sol.g(↩
   n_states*(N+1)+(n_veh_safty_states*N)+(n_pred_len*↩
   N)+no_of_obs_detect*(N+1)+1 : n_states*(N+1)+(↩
   n_veh_safty_states*N)+(n_pred_len*N)+↩
   no_of_obs_detect*(N+1)+n_mvobs*(N+1)*(N/mv_fact)))↩
   );
pred_state_matrix1 = reshape(full(sol.x(1:n_states*(↩
   N+1)))',n_states,N+1);
t(mpciter+1) = t0;
[t0, x0, u0] = shift_mvobs( dt, t0, x0,↩
   obs_velo_update,obs_theta_update, u, slope_rad, ↩
   model_fn);  % use this
[mv_obs_velo_ms,mv_obs_theta_rad,mv_obs_pos_xy] = ↩
   mv_obs_fn_adv(x0,u(1,:)',slope_rad(1),↩
   mv_obs_velo_ms,mv_obs_theta_rad,model_fn,obs_coord↩
   ,obs_velo_ms,obs_theta_rad,dt); % was u(1,3), dt
x0((n_states-n_mvobs_states)+1:end) = mv_obs_pos_xy;↩
    % update the position (x,y) of moving obstacle,
% update the obstacle-1 and 2 information
for i_1 = mvobs_ind
     obs_coord(1,i_1) = obs_coord(1,i_1) + ↩
        obs_velo_ms(i_1)*cos(obs_theta_rad(i_1))*dt;  ↩
        % was u(1,3), dt
    obs_coord(2,i_1) = obs_coord(2,i_1) + ↩
       obs_velo_ms(i_1)*sin(obs_theta_rad(i_1))*dt;  ↩
       % was u(1,3), dt
end
obs_pos = reshape(obs_coord,(no_of_obs*obs_states)↩
   ,1);
[rpm,T_eng,effi_curr] = LVD_model(x0(4),u(1,1));
con_fn_value =con_fn(x0);      % use current states ↩
   and control inputs and calculate ODE fn value
% bounds on control inputs
args.lbw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(2));   % lower bound on acce, (Use ↩
   eqn s for it)
args.ubw(7:n_states:n_states*(N+1),1) = full(↩
   con_fn_value(1));
x0(3) = hdng_mapping(x0(3));
[tar_hdng,x_ref(4),obs_info,slope_deg] = ↩
   box_obs_slope_contour_search(x0(1:3),x_tar(1:3),↩
   obs_coord,x_terr_map,y_terr_map,z_terr_map);
if const_spd_case
    x_ref(4) = 20;  % make simulation with costant ↩
       speed of 20 m/sec
end
if ~isempty(obs_info)        % (length(obs_info)) > 0
    obs_info_out = [obs_info_out',obs_info];
    obs_info_out_coord = reshape(obs_info_out,2,↩
       length(obs_info_out)/2);
    [UniXY,Index]=unique(obs_info_out_coord','rows')↩
       ;
    obs_info_out = reshape(UniXY',(size(UniXY,1)*2)↩
       ,1);
    % delete the stored obs_data, that is more than ↩
       125 m from the curr vehi position,
    for i = length(obs_info_out)/2:-1:1
```

```
        dist_m = sqrt((obs_info_out(2*i-1) - x0(1))↩
           ^2 + (obs_info_out(2*i) - x0(2))^2);
        if dist_m > 125
            obs_info_out((2*i-1):2*i) = [];
        end
    end
    if ~isempty(obs_info_out)
        for i = length(obs_info_out)/2:-1:1
            % let's remove the obstacle from storage↩
               , if it is moving  condition for ↩
               moving is,
            for j = length(mv_obs_pos_xy)/2:-1:1
                dist_m1 = sqrt((obs_info_out(2*i-1) ↩
                   - mv_obs_pos_xy(2*j-1))^2 + (↩
                   obs_info_out(2*i) - mv_obs_pos_xy↩
                   (2*j))^2);
                if dist_m1 < 0.9
                    obs_info_out((2*i-1):2*i) = [];
                end
            end
        end
    end
    if (length(obs_info_out)) <= n_obs_detected
        detected_obs_pos(1:length(obs_info_out),1) =↩
            obs_info_out;
    else
        detected_obs_pos = obs_info_out(1:↩
           n_obs_detected,1);
    end
end
x_ref(3) =  tar_hdng;
slope_rad(1) = slope_deg(1)*pi/180;
slope_rad(2) = slope_deg(2)*pi/180;
```

```
    xx(:,mpciter+2) = x0;    % take the history of sates ↩
       and stores it into this parameter
    ux(:,mpciter+2) = u(1,:);    % take the history of ↩
       control inputs and store it in this parameter
    veh_sfty_N(:,mpciter+2) = veh_sfty(1,:);
    obs_his(:,mpciter+2) = obs_pos; % take the history ↩
       of obstacle information
    detected_obs_pos_his(:,mpciter+2) = detected_obs_pos↩
       ; % for plotting the obstacles detected along the ↩
       way
    detected_mov_obs_pos_his(:,mpciter+2) = ↩
       mv_obs_pos_xy;
    detected_mov_obs_velo_his(:,mpciter+2) = ↩
       mv_obs_velo_ms;
    detected_mov_obs_theta_his(:,mpciter+2) = ↩
       mv_obs_theta_rad;
    vehi_oper_pts(:,mpciter+2) = [effi_Tar;rpm;T_eng;↩
       effi_curr];        % for making efficiency plots
    pred_state_mat_init = reshape(full(sol.x(1:n_states↩
       *(N+1)))',n_states,N+1)';
    pred_state_mat_init = [pred_state_mat_init(2:end,:);↩
       pred_state_mat_init(end,:)];
    hdng_err_mat_init = hdn_err_fn(pred_state_mat_init,↩
       x_ref(3));
    mpciter
    mpciter = mpciter+1;
end
main_loop_time = toc(main_loop);
ss_error = norm((x0(1:3)-x_ref(1:3)),2)
average_mpc_time = main_loop_time/(mpciter+1)
```

B.3 ABD-JDN algorithm flow chart



Flow Chart: ABD-JDN Obstacle-Grade Avoidance Algorithm
Make Obs-Grade search in Striaght section (Boxes [1,-1])
No obstacle and No grade obstacle
yes
Execute/call Avoidance algorithm for Straight section
No
Right-(i) section search
i=(i+1)
No obstacle and No grade obstacle
No
Yes
Execute/call Avoidance algorithm for Right section
i-loop
i < max sections
Go Extreme Right section
Yes
Left-(i) section search
No
No obstacle and No grade obstacle
Yes
Execute/call Avoidance algorithm for Left section
Straight Section search(Both obstacles and flat road)
Execute/call Avoidance algorithm for Straight section
road grade < Llimt
Go Straight
slope region in st. section
Make search in Right-1 section (Boxes [2,3])
Go Straight with slope region
No obstacle and No grade obstacle
No
Yes
road grade < Llimt
Go Right-1 section
Slope region in Right-1 section
Make search in Left-1 section (Boxes [-2,-3])
Chose the min slope region between st. and Right-1 section
No obstacle and No grade obstacle
No
Yes
road grade < Llimt
Go Left-1 section
Slope region in Left-1 section
Chose the min slope region between st., Right-1 and Left-1section
Vehicle Avoidance Algorithm - Right side
Execute/call Avoidance algorithm for Right section
road grade < Llimt
Go Right-(i) section
slope region in Right-(i) section
i < max sections(3)
No
Go Right-(i) section
Yes
Make search in Right-(i+1) section
No obstacle and No grade obstacle
No
Go Right-(i) section
Yes
road grade < Llimt
Go Right-(i+1) section
Slope region in Right-(i+1) section
i < max sections
No
Chose the min slope region between Right-(i) and Right-(i+1) section
Yes
Make search in Right-(i+2) section
No obstacle and No grade obstacle
No
Chose the min slope region between Right-(i) and Right-(i+1) section
road grade < Llimt
Go Right-(i+2) section
Slope region in Right-(i+2) section
Chose the min slope region between Right-i, Right-(i+1) and Right-(i+2) section
Vehicle Avoidance Algorithm - Left side
Execute/call Avoidance algorithm for Left section
road grade < Llimt
Go Left-(i) section
slope region in Left-(i) section
i < max sections(3)
No
Go Left-(i) section
Yes
Make search in Right-(i+1) section
No obstacle and No grade obstacle
No
Go Left-(i) section
Yes
road grade < Llimt
Go Left-(i+1) section
Slope region in Left-(i+1) section
i < max sections
No
Chose the min slope region between Left-(i) and Left-(i+1) section
Yes
Make search in Left-(i+2) section
No obstacle and No grade obstacle
No
Chose the min slope region between Left-(i) and Left-(i+1) section
road grade < Llimt
Go Left-(i+2) section
Slope region in Left-(i+2) section
Chose the min slope region between Left-i, Left-(i+1) and Left-(i+2) section

Figure B.1: Flow chart for the ABD-JDN algorithm.

Appendix C

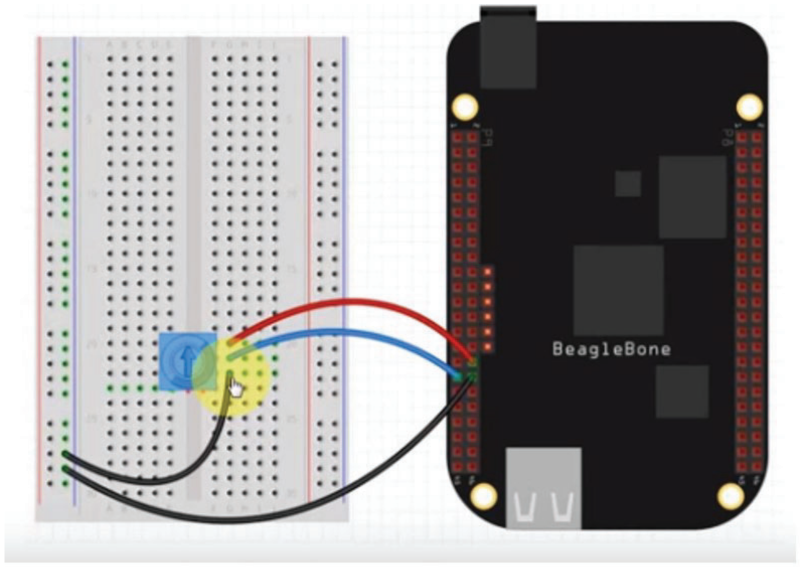
Path tracking algorithms

C.1 Example of Analog read on Beagle bone black

The below code and schematic provide basic example on how to operate Beagle bone balck embedde system using python script. The Fig .C.1 shows the connections between potentiometer and Beagle bone black analog pins. The sensor mimics the steering measurement and the fixed gear ratio provide the steering wheel angle of 1/5*th* truck. Application: For steering sensor/pot sensor reading

C.2 Steering sensor measurement code

```
% Analog Read code for steering sensor measurement
import Adafruit_BBIO.ADC as ADC % we need this library ↩
   to use ADC pins
from time import sleep % we need this for having delay ↩
   during measruments
ADC.setup() % this line activates all the ADC pins on ↩
   BBBw
analogPin = "P9_33" % assign 33 pin to analogPin ↩
   variable
```



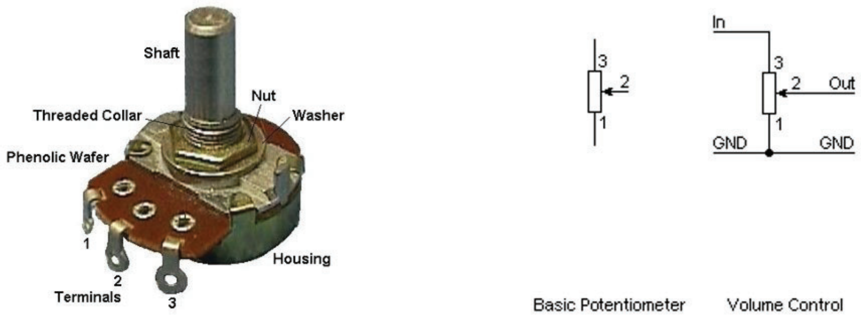


Figure C.1: Potentiometer interface with Beagle bone black

```
while(1):
    currVal = ADC.read(analogPin) % usedADC.read cmd to ↩
       read sensor value
    print "The current Value is: ", currVal
    sleep(0.2)
```

C.3 Stanley method

This method was developed by DARPA team for tracking the path using simple steering angle relation [59]. The steering commands for navigating the vehicle can be calculated based on the cross track error calculated from vehicle kinematics model with respect to the desired path is shown in Figures C.2 and C.3. The control gain

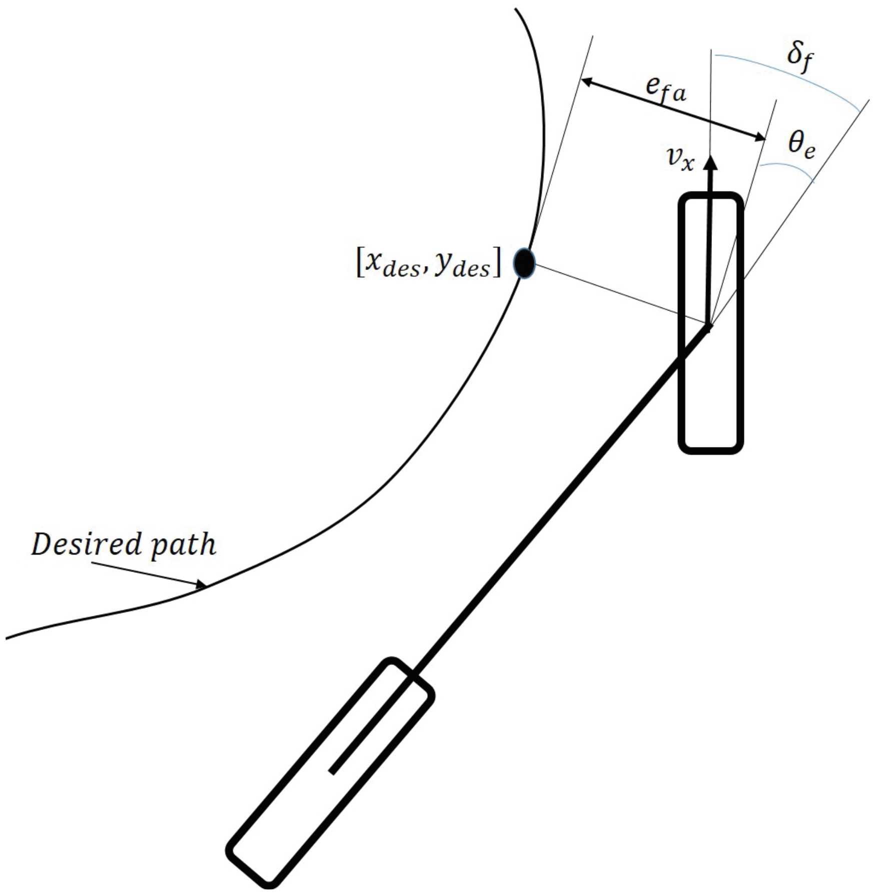


Figure C.2: Vehicle bi-cycle model along the desired path coordinates

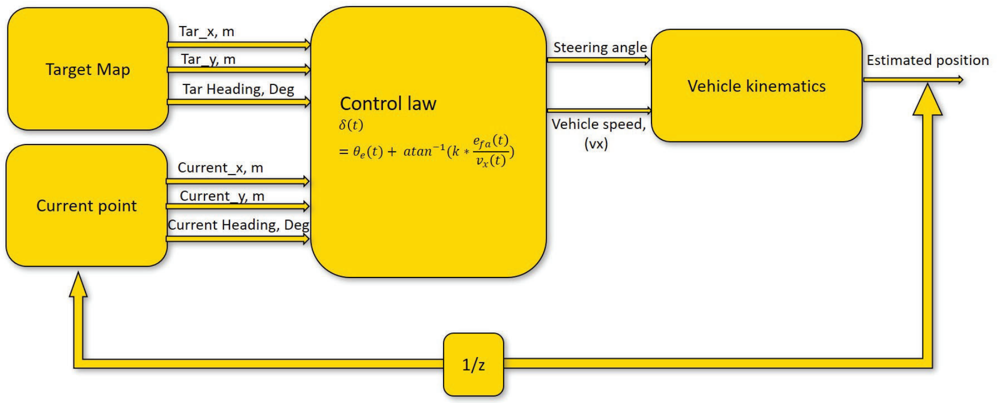


Figure C.3: Schematic of Stanley control law

used in the present test conditions is 0.15. The initial simulations were made with the above conditions and results shows the good agreement with test data.

Table C.1
Sensor properties used in LQG controller development.

Sensor	Application	specifications	Range	resolution	maker	chipset
GPS Base station	Position measurement	GNSS, 72 channel, Tracking:=-164 dBm,5hz update	2.5 km	$\pm 0.35m$	Inertial sense	EVB-2
Gyro	rate measurements	$1 - 255hz$ update	$\pm$2000 deg/s	$\pm$0.06 deg/s/-rms	Chrobo tics	UM7
Accelero meter	acceleration measurement	$1 - 255hz$ update	$\pm 8g$	400μ g/rthz	Chrobo tics	UM7
Magneto meter	heading measurement	$1 - 255hz$ update	$\pm$1200 μT	$\pm$4%	Chrobo tics	UM7

C.4 Sensor properties

Table. C.1. Provides the various sensor properties used in the LQG controller development.