SYSTEM IDENTIFICATION OF AUTONOMOUS ON AND OFF-ROAD VEHICLES WITH NON-LINEAR MODEL PREDICTIVE CONTROL FOR PATH TRACKING

A Thesis

by

WAEL KARKOUB

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2021

Major Subject: Mechanical Engineering

ABSTRACT

There is no question that autonomous vehicles and robots are going to have a profound impact on the society in the near future. A deep understanding of the systems is required in-order to design safe autonomous systems that are deployed among people. A nonlinear data-driven system identification techniques was explored to improve the physics-based models coupled with a nonlinear MPC for path tracking for the Warthog and ASV. A novel approach for a longitudinal controller was also introduced for ASV. Domain knowledge was extensively used to generate a feature set for the machine learning algorithms to learn. During the experiment, it was found that for the Warthog dataset, it needed at least 103 seconds worth of data to outperform the physics-based model. No conclusion can be made regarding the generality of the model. The learned models, for both the Warthog and ASV, returned the physics-based model with the parameters identified, suggesting that there is a slight delay in the system that was not accounted for given the nominal values. The nonlinear MPCs were validated using the Gazebo and CARLA simulators. The MPC for the Warthog followed the predefined path with an average error of 8 cm, however, it struggled with sliding while turning. The MPC for the ASVs were tested using three different scenarios; highway exits and entrances, highway driving, and city driving. Both the lateral and the controller tracked the generated path within the safety margin, which is the lane width. However, it failed the city driving test, exceeding the cross-track error of 3 m.

DEDICATION

To my family.

ACKNOWLEDGMENTS

First, I would like to thank my advisor Dr. Srikanth Saripalli whose efforts to steer me in the right direction formed the foundation to my graduate studies. Furthermore, I would like to acknowledge the respectable members of my research committee, Dr. Dylan Shell and Dr. Alan Palazzolo, for their valuable insights and consistent support. The time I've spent working at the Unmanned Systems Lab has been rich with academic and personal experiences that helped me grow on many levels. For that, I am grateful to every member of this inspiring team. Collaborating with them has made this journey even more worthwhile.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Srikanth Saripalli and Professor Alan Palazzolo of the Department of Mechanical Engineering and Professor Dylan Shell of the Department of Computer Science and Engineering.

The Warthog's Odometry is setup by Unmanned Systems Lab members, Tim Overbye and Akhil Nagariya. All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University and a dissertation research fellowship.

NOMENCLATURE

ASV	Acekermann Steering Vehicle
MPC	Model Predictive Control
RLS	Recursive Least Squares
GNSS	Global Navigation Satellite System
INS	Inertial Navigation System
EOM	Equations of Motion
COR	Center of Rotation
CC	Cruise Control
OCP	Optimal Control Problem
CTE	Cross-Track Error
GPS	Global Positioning System
LiDAR	Light Detection and Ranging
ROS	Robotic Operating System

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

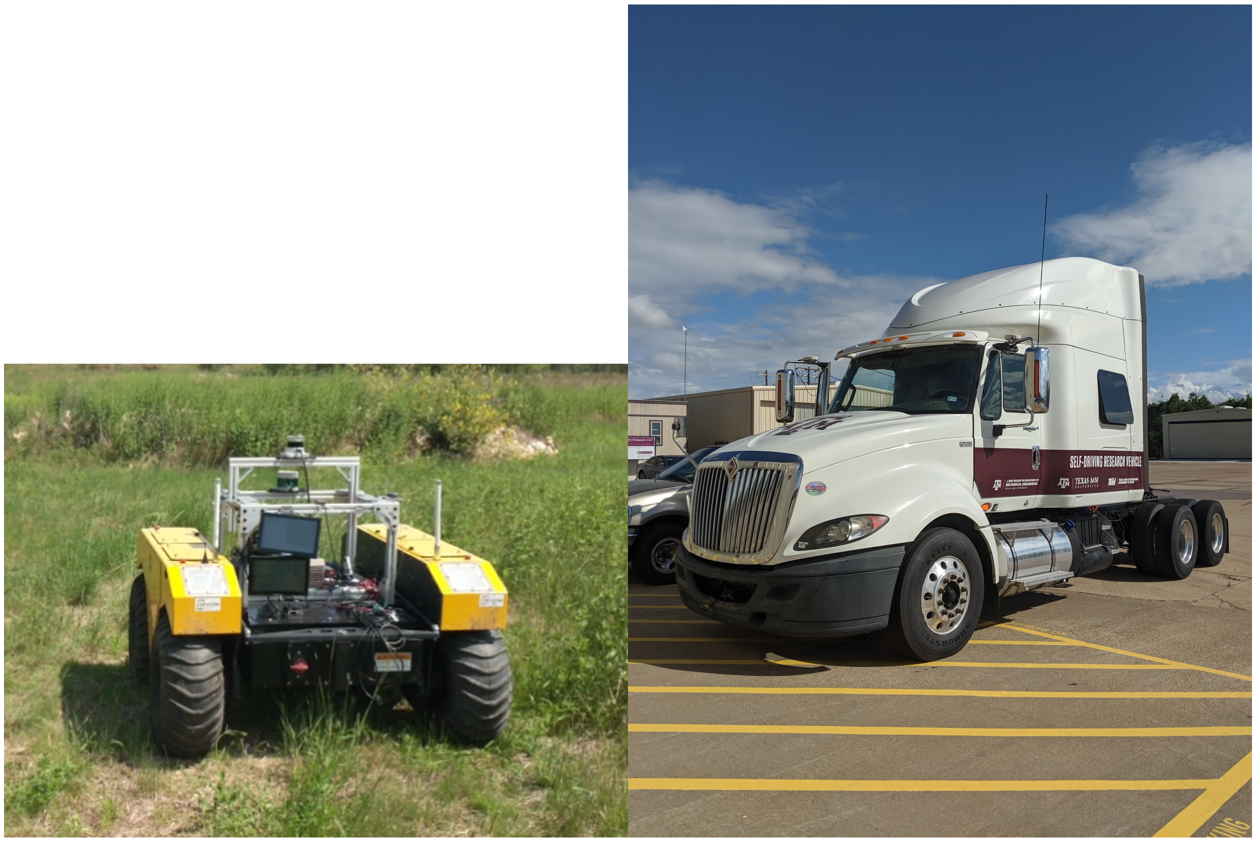
TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Recent advancements in sensor and compute technologies allowed researchers to shift focus towards autonomous systems that better the lives of people. Innovation in autonomous systems allowed for safer roads, efficient logistics, higher security, and deeper exploration of land, sea, and space. Technology companies these days are allocating most of their research resources towards developing autonomous solutions that can be used in transporting goods in warehouses, transporting people from one to place to another, and also security applications such as border patrol. However, this recent surge in interest did not start out of nowhere. DARPA Grand Challenge [1] started in 2004, allowed teams to develop vehicles to cross the Mojave Desert autonomously and compete for a cash prize. The success of the Grand Challenge encouraged more research to be conducted in software development for autonomous systems in perception, controls, and state estimation. The major issue that autonomous systems face is the ability to function at a high level of safety in presence of other dynamic elements such as other people and robots.

This thesis tackles two major challenges that is faced by every autonomous system design. The first challenge is modelling the system of interest. A model is a mathematical representation of the system. In this thesis, various approaches to system modelling are explored to overcome common issues faced such as inaccuracies in physics based models or difficulties in obtaining a physics based model. Obtaining a good model is important as it is used for simulating and controlling the system, which leads to the second major challenge. The purpose of the controller is to induce a desired behavior of the system by applying inputs onto the system.



(a) (b)

Figure 1.1: (a) Warthog: amphibious unmanned ground vehicle developed by ClearPath Robotics. (b) Autonomous semi-truck in Unmanned Systems Lab, an example of an Ackermann Steering Vehicle .

In this thesis, a lateral and longitudinal model predictive controllers (MPC) will be designed for and implemented on two different systems; an amphibious unmanned ground vehicle called Warthog (Figure 1.1a) that utilizes differential-drive steering mechanism, and for an ackermann steering vehicle (ASV), seen in Figure 1.1b.

1.2 Literature Review

1.2.1 System Identification/Estimation

The system model must be able describe the vehicle's behavior without being too computationally intensive [2]. Computationally intensive models may lead to more accurate results, however, it will lead to slower times to solve the optimization problem for the MPC. Solving the optimization problems is required to be faster than the actuation's sampling rate or it might introduce unwanted input lag. Accurate models, however, can be used for simulation as there is no optimization process required. This section introduces some of the published work on system modelling and parameter estimation.

Kong et al. [3] studied the differences between a kinematic and dynamic models for ASVs that will be later used for their MPC. They have also studied the effect of sampling rate on the accuracy of the prediction model. The dynamic model used a linear tire model. The research shows that the kinematic model discretized at 200 ms performs as well as or better than the dynamic model discretized at 100 ms. However, the authors noted that the dynamic model may be better suited for high-speed driving and high lateral acceleration turns. Ercan et al. [4] proposed a data-driven approach to model the steering system. The authors have used a recursive least squares (RLS) algorithm as opposed to a common steering model. One of the benefits of using RLS is the ability to adapt to changing conditions, such as road surface and vehicle's condition. The results show that the RLS model outperformed the conventional steering model by improving the stability of the vehicle while using less control inputs.

Despite the fact that the dynamic models are more representative of the vehicle's behavior, parameters such as cornering-stiffness and tire-friction coefficients are difficult to obtain due to changing road conditions and tire degradation over-time. Hu et al. [5] proposed a Kalman filter framework approach to estimating these parameters at real-time. The framework implements an extended Kalman filter to estimate the lateral velocity of the vehicle, and then used two unscented Kalman filters cascaded to estimate the friction force and then estimate the tire-friction coefficient. The proposed method resulted in accurate and fast estimation of the tire-friction coefficients. The authors also showed that this method worked better for double-lane change scenarios when compared to single lane change.

1.2.2 Geometric and Rule-Based Controllers

Geometric controllers applies the ideas of differential geometric techniques to control theory. These controllers have proven stability for robotics applications. The benefit of having geometric controllers is that it is simpler to calculate when compared to optimal controllers. Therefore, allowing it to run efficiently on weak compute units. Rule-Based controllers, as the name suggests, are controllers that are based on rules such as IF-ELSE rules. A popular example of a Rule-Based controller would you be Fuzzy Logic controllers. This section introduces some of the published work on geometric and rule-based controllers.

One of the most popular example of Geometric controllers is the Stanley controller [6]. Hoffman et al. detailed the work of a nonlinear lateral controller for ASVs that has proven stability. A typical lateral controller for path tracking takes into account the orientation of the vehicle, however, the authors took a different approach. They considered the angle between the front wheels and the reference path. This novel controller design allowed them to win the DARPA Challenge in 2005. Other researchers have expanded the work of Hoffman et al. by adding predictive capabilities [7] or improving the controller using fuzzy logic [8].

Another example of a Geometric controller is the Pure Pursuit Controller [9]. Coulter et al presented a report that described the implementation of the pure pursuit path tracking algorithm as it showed promise as tracking algorithm for land-based navigation. The Pure Pursuit algorithm works by calculating the curvature between two points; the position of the robot and the goal point. This is done by creating an arc, called the “look-ahead distance.” The advantages of the Pure Pursuit Controller is that it is computationally efficient and stable for the Warthog and ASVs.

1.2.3 Optimal Controllers

Optimal controllers is an optimization problem in which the goal is to obtain a control law given a cost function and system constraints. There are many examples of optimal controllers such as LQR, LQG, H_2, MPC, etc. Only MPCs will be discussed in this section as it is the focus of the thesis. MPC is an optimal controller that optimizes over a finite horizon given a set of trajectory references, system dynamics, and system constraints. An example of a trajectory references would be the path and the velocities along the path that the Warthog or ASV needs to track. The advantages of MPC are the ability to look-ahead in time and optimize accordingly, and it is inherently stable according to Pannocchia et al. [10]. This section introduces some of the published work on MPCs.

Ercan et al. [4] proposed a control framework for ASVs in which improved tracking and stability under different road conditions, such as dry, wet, and icy roads. Their main contribution was implementing the learned RLS steering model into the vehicles dynamics. The results show that by accounting for steering dynamics, the MPC was able to stabilize the vehicle without overshooting the trajectory. In recent years, there is an increase interest for learning based controllers such Rosolia and Borelli [11]. They presented a Learning Model Predictive Control framework that guarantees performance improvement at each iteration. Hewing et al. [12] reviewed recent work on learning-based MPC, stochastic MPC, and implementing safety constraints in MPCs. A novel approach presented by Wang et al. [13], combined deep-learning, a growing deep belief network (GDBN), with an optimal controller. Calling it the DeepMPC, proved to have better disturbance rejection and better modelling accuracy when compared to the conventional MPC designs when applied to a continuous stirred-tank reactor.

1.3 Autonomous System Architecture

1.3.1 General System Architecture

There are a myriad of ways to designing a good software and hardware architectures for autonomous systems. A general autonomous system architecture that is widely accepted by the robotics community will be described in this section [14], shown in Figure 1.2. The overarching idea behind designing a good autonomous system architecture is dividing the entire system into subsystems, or modules. The benefit of modularizing the system is it makes it easier to develop and maintain modules without worrying about effecting other modules, as the input and output for each module is strictly predefined.

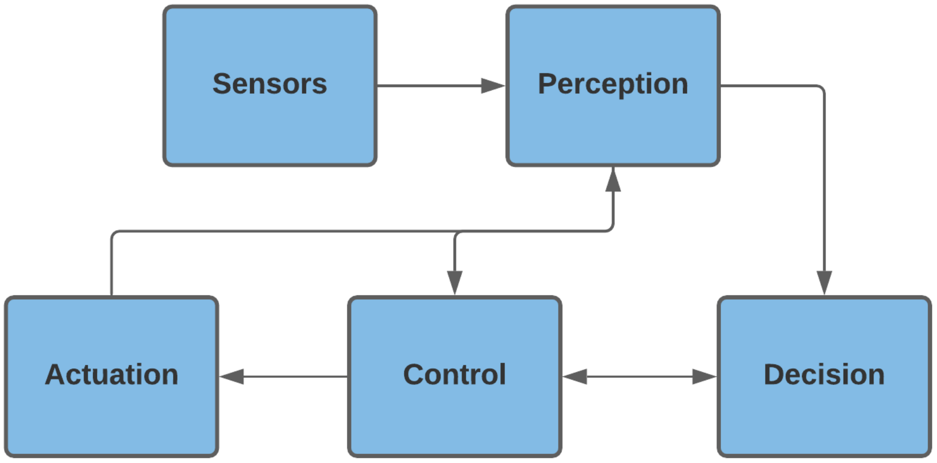


Figure 1.2: Diagram of the software/hardware architecture for autonomous systems.

The first main module is the sensor stack, which may include cameras, GNSS, INS, radar, LiDAR, wheel encoders, etc . The goal of the sensor module is to obtain the state of the system and actors in the environment. For example, the wheel encoders could be used to obtain the velocity of the robot while the LiDAR can be used to determine if there are obstacles in the path. Note that filtering of the data can also be done in the sensor module especially if the filtering is done at the hardware level, however, some filtering can be done in perception module which will be discussed next.

The second main module is the perception module, which is responsible of localizing the robot and the obstacles around it. Some applications require the process of mapping, which is also done in the perception module. The perception module should be able to receive data for the sensor module to perform its task.

The third module is the decision module, which is responsible for planning the missions for the robot. Since this thesis focuses on the Warthog and ASVs, the planning tasks is divided into two sub-tasks; global and local planning. The global planner is responsible for providing the overarching goal to the robot, such as move an item from point A to point B, while the local planner provides the exact path the Warthog or ASV must follow. Local planners take into account the state of the vehicle as well as obstacles.

The fourth module is the control module, which is the focus of this thesis. The purpose of the controller is to provide input to the system to comply with what the local planner has provided. The control module should obtain the state of the vehicle, and depending on the controller design, the obstacles from the perception module as well. In the case of the Warthog and ASVs, the controller should obtain the path, known as waypoints, and the velocity at each point of the path. The controller then sends signals to the actuation module to perform the desired actions.

1.3.2 Control Module Architecture

This section will discuss the Control module in more detail, shown in Figure 1.3. The first main module is the system model. It receives iniformation such as odemtry and vehicle data, clean it, extract features, and fits a regression model. Once the system model is obtained, it feeds the model to the controllers. There are two controller types that this thesis will focus on; High-level and low-level controller. The high-level controller is responsible for controlling the state of the vehicle, such as the speed and the position. While the low-level controller is responsible for mapping the output of the high-level controller to actuation. This separation is important as it allows to design different controllers for systems that are difficult to model such throttle/brake models for ASVs.

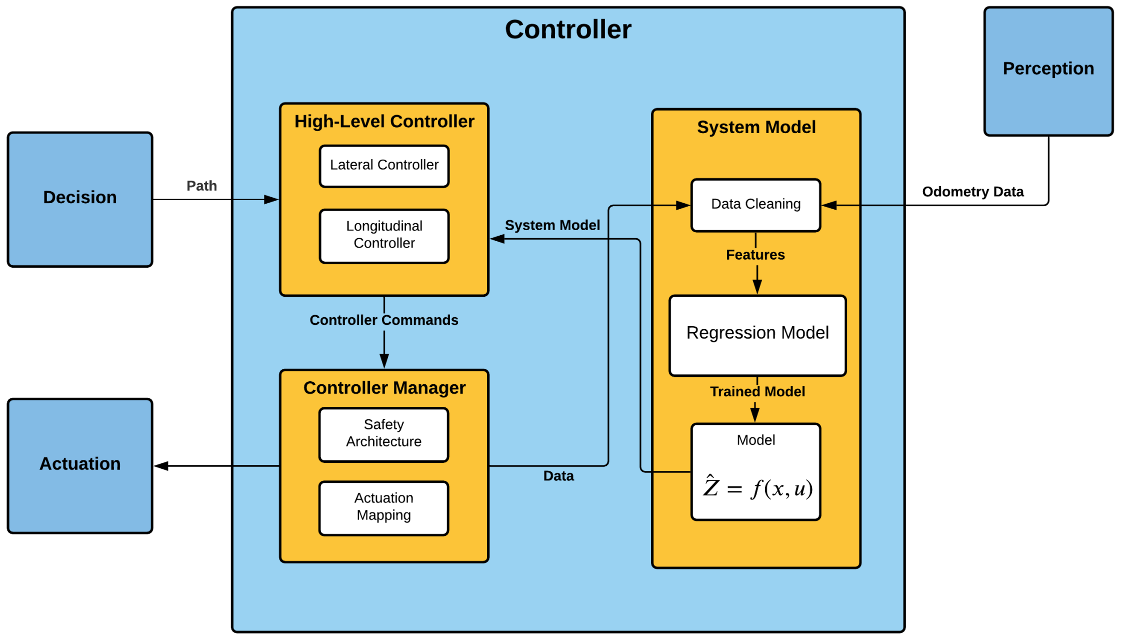


Figure 1.3: Diagram of the Control Module.

The controller manager acts as the safety controller for the autonomous system in-case of a failure of one of the controllers. The controller manager should have to access to optimal and geometric controllers at any given time, allowing it to switch between the two if any one of them fails.

1.4 Objectives

The individual contributions resultant from this thesis are itemized below:

- Derive system models for the Warthog and ackermann steering vehicles (ASV) using a data-driven approaches using different feature engineering methods.
- Compare different linear and non-linear machine learning (ML) approaches for system identification.
- Develop non-linear MPC for ASVs, that is capable of driving in cities and highways.

- Develop a throttle/brake controller using a novel learning based method with a the summation of PID acting as a correction term in-case of downhills and uphills.
- Develop and implement a non-linear MPC for the Warthog for off-roading capabilities.
- Develop a simulator to account for input uncertainties and delays for ASV and differential-drive robots.
- Develop a ROS wrapper for CARLA to emulate PACMOD and Vectornav.

1.5 Overview

The following chapters will introduce major concepts that are needed to design a data-driven system identification and controllers. Chapter 2 will introduce the concept of feature engineering and how it can be used to come up with a model. It will discuss how to extract features from domain knowledge to improve the models. Then the concepts of linear and nonlinear predictors will be introduced and derived. Chapter 3 will introduce the major concepts to design nonlinear MPCs. Optimal controls is formulated and illustrates how MPCs fit in that family of controllers. Then optimization problems will be defined for the Warthog and the ASV. Additionally, a novel longitudinal controller will be derived used a data-driven approach. Chapter 4 will introduce the different simulators used to validate the work done in this thesis. The system identification techniques introduced in chapter will be evaluated. The controller designs will also be evaluated for the Warthog and ASV. Finally, the last chapter will conclude the work done in the thesis and discuss future work to improve upon this thesis.

2. SYSTEM MODELING

The performance of the MPC highly relies on having a good model of the system. The MPC uses the model to predict the future states of the vehicle. Therefore the system model must describe the vehicle dynamics accurately. One significant trade-off while modeling the system is considering the computational complexity of the model. Having an accurate model while the optimizer is in-capable of obtaining the solution in time can worsen the controller's performance by adding unwanted delay. This chapter will introduce physics-based models for differential-drive robots and Ackermann steering vehicles, different approaches to feature engineering for modeling purposes, and introducing algorithms for linear and nonlinear predictors.

2.1 Feature Engineering

The first important step of any system identification is understanding what factors that effects the behavior of the system. This step is generally known in the data science community as feature engineering [15]. factors and features can be uses interchangeably. The features that the engineer selects for the model has direct effect on the accuracy and quality of the predictive model. In other words, the better the features, the better is the model. Another benefit of selecting good features is it also translates to simpler model, which will enhance the computation efficiency of the MPC.

There is no correct way of feature engineering as its more generally regarded as much of an art as science. However, there are few tools and techniques that can guide an engineer to which features are of importance. The first category of technique is called Filter Methods. Chi-square, information gain, and the correlation coefficients are all examples of filter methods; they use statistical techniques to score each feature based on their correlation between each of the features and the statistical significance to the predictive power. The second technique is called Wrapper Methods, which includes Forward Feature Selection, Backward Feature Elimination, and Exhaustive Feature Selection, These techniques use and iterative based method to determine which of the features are important to the prediction model. At each iteration, the system removes or adds a feature and determines if said feature is needed to improve the model. The last technique, which is the focus of this thesis, is called Embedded Methods or Regularization. Regularization is when a penalty term added in the objective function of the predictor to penalize the coefficients. This approach is more computationally efficient since the training process of the model accounts for the feature selection.

There are two main types of regularization; L_1 and L_2 [16]. L_1 encourages the coefficients of the model to approach to zero. Having L_1 in the objective function will eliminate any feature that does not improve the prediction model.

$$L_1 = \sum_{i=0}^{p} |\beta_i| \tag{2.1}$$

Where β_i are the coefficients of the model. L_2, on the other hand, encourages the coefficients to approach zero as close as possible but can never reach zero like L_1. The goal behind L_2 is to avoid any over fitting of an important feature by penalizing its coefficient.

$$L_2 = \sum_{i=0}^{p} \beta_i^2 \tag{2.2}$$

The next step is to select predictors that utilizes both L_1 and L_2 in their objective function, which will be discussed later in the section.

So how would one generate features that can describe the system? One popular method, and is the recommended method when possible, is to use domain knowledge. For this thesis, the domain knowledge is obtained using the physics-based model for the Warthog and ASV. By using the equations of motion (EOM), the engineer can use the equations as part of the features used to train the predictive model.

2.2 Kinematic System Models

Selecting the appropriate models for the controller falls back to the assumptions made about the system. The following assumptions are:

- The states of the vehicles must be directly measured,

- the vehicles stay below the tires' lateral and longitudinal limits,
- no-slip,
- vehicle's motion is on a 2D plane,
- and computationally in-expensive.

Given the assumptions, kinematic models are ideal for MPC as they provide all the necessary vehicle's state information (position and heading), neglect tire-road interactions, and can be executed efficiently. Although dynamic models are more accurate as they consider the system's inertia, some model parameters are difficult to obtain, such as the tire-road coefficient.

2.2.1 Differential-Drive

The majority of mobile robots, such as the Warthog, depend on the differential drive for steering due to its simplicity. The mechanism hinges on having two independently controlled wheels placed on each side of the robot on the same axis; the direction and the speed of the robot depend on the rotational speed differences and the average speed of both wheels, respectively. If both wheels rotate at the same speed, then the robot will move in a straight line. If the wheels rotate at different rates, then the robot will rotate about the center of rotation (COR). The radius away from COR depends on the wheels' rotational speeds, which will be discussed further later on.

By controlling the speeds of each wheel, the trajectory of the robot can also be controlled. Since both wheels lie on the same axis, the rotation rate for at each wheel about COR must be the same [17], therefore:

$$\omega(t) = \frac{v_r(t)}{COR + \frac{L}{2}} \tag{2.3}$$

$$\omega(t) = \frac{v_l(t)}{COR + \frac{L}{2}} \tag{2.4}$$

Where L is the wheelbase of the robot, v_r and v_l are the tangential velocities of the right and left wheel, respectively. By subtracting Eq. 2.4 from Eq. 2.3, the rotational speed of the robot can

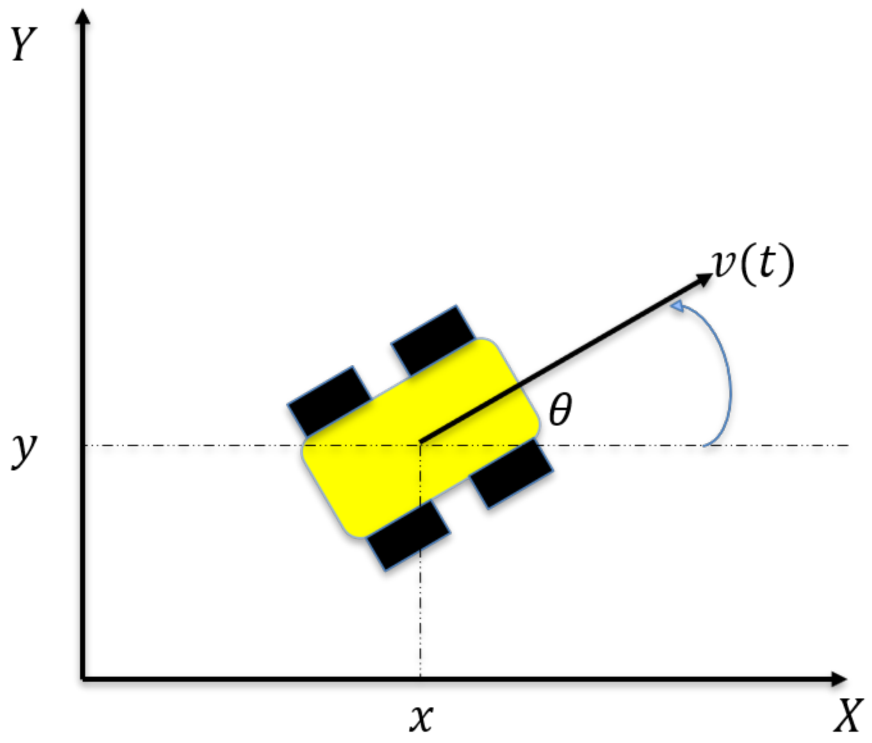


Figure 2.1: Differential-Drive model diagram.

be formulated as :

$$\omega(t) = \frac{v_r(t) - v_l(t)}{L} \tag{2.5}$$

while averaging the tangential speed of both wheels yields to the tangential speed of the robot:

$$v(t) = \frac{v_r(t) + v_l(t)}{2} \tag{2.6}$$

Given the robot's position as x and y, and the global frame as X and Y, the equations of motion (EOM) for a differential-drive robot are described in equations 2.7.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix} \tag{2.7}$$

MPCs require the models to be discretized. Therefore, the EOM will be discretized using zero-order hold estimation.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} x + v\cos(\theta)\Delta t \\ y + v\sin(\theta)\Delta t \\ \theta + \omega\Delta t \end{bmatrix}_{k} \tag{2.8}$$

Where Δt is the sampling rate of the controller. As discussed earlier, we can generate features for the prediction model using the obtained equations. If we follow the approach Ercan et al. [4] suggested, a linear combination of all states and inputs would be used as a model, and example would look like:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \alpha_0 + \alpha_1 x_k + \alpha_2 y_k + \alpha_3 \theta_k + \beta_0 v_k + \beta_1 \omega_k \tag{2.9}$$

Where α_i and β_i are the coefficients. The major downside of this approach is that it does not capture the interactions between all the states and inputs, therefore valuable information is expected to be lost. The author of this thesis suggests a polynomial nonlinear combination of all nonlinear states and inputs of the system:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{h=0}^{n}\sum_{l=0}^{n}\sum_{m=0}^{n}\sum_{q=0}^{n} W_{i,j,h,l,m,q} x_k^i y_k^j \cos^h\theta_k \sin^l\theta_k v_k^m \omega_k^q \tag{2.10}$$

Where W are the coefficients of the model. By multiplying the states and the inputs, the interaction behaviour is captured.

2.2.2 Ackermann Steering

2.2.2.1 Vehicle State Model

The bicycle model is the simplest kinematic model that captures the behavior of a front-steering vehicle. The model reduces a 4-wheel model to a 2-wheel model by combining all wheels that are on the same axle into a single wheel. In other words, the front tires are combined to account for only one steering angle. The concept that allows the bicycle model to be a good representation for ASVs is due to the COR of the vehicle. If you extend a perpendicular line from any point on an ASV with respect to their velocity vector, they intersect at the COR, assuming no-slip, of course. Figure **??** illustrates the instantaneous COR. By comparing the 4-wheel model to the bicycle model, it can be seen that both models have the same COR if the steering angle of the bicycle model was equal to the average of steering angles of the 4-wheel models. This model simplification yielded a faster controller law solution and system identification.

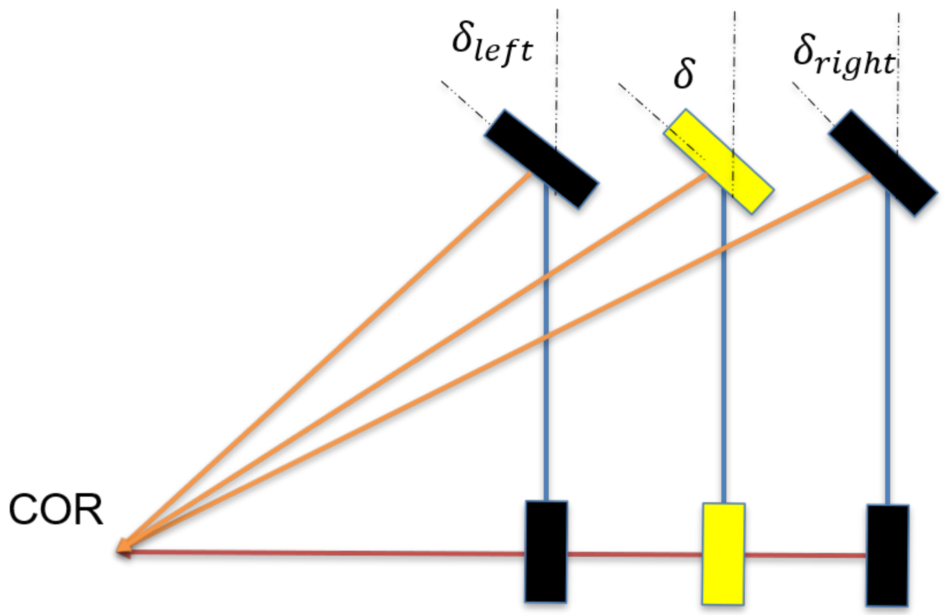


Figure 2.2: Diagram to illustrate the COR of an ASV.

The bicycle model states are the position on a 2d plane and the heading angle of the ASV, and

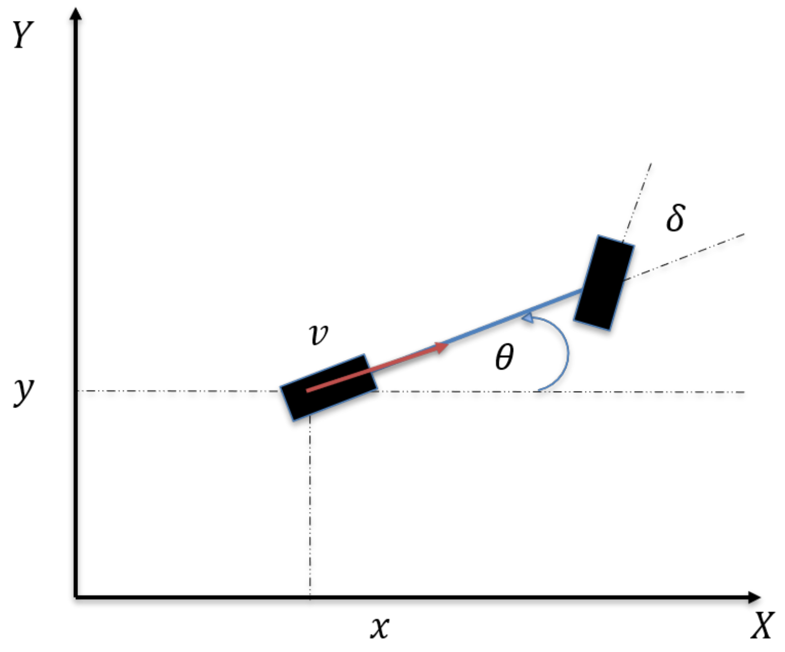


Figure 2.3: Ackermann steering model diagram.

the inputs to the model are the acceleration of the vehicle and the steering angle. Before deriving the EOM, a reference point (RP) must be selected at a point in the vehicle. Three commonly selected points are at the front axle, rear axle, and at the center of gravity (COG). Selecting the RP to be COG is not ideal in most cases as it can change over time due to load change, such as when the number of passengers changes. Selecting the RP at the front axle and the rear axle, in terms of derivation, are similar. Therefore, this thesis will move on with deriving the EOM of an ASV with the rear axle as an RP.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} x + v\cos(\theta)\Delta t \\ y + v\sin(\theta)\Delta t \\ \theta + \frac{v}{L}\tan(\delta)\Delta t \end{bmatrix}_{k} \tag{2.11}$$

Where δ is the steering angle of the vehicle, and a is the acceleration of the vehicle. Using the nonlinear approach to feature extraction, the new equations will look as follows:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{h=0}^{n}\sum_{l=0}^{n}\sum_{m=0}^{n}\sum_{q=0}^{n} W_{i,j,h,l,m,q} x_k^i y_k^j \cos^h(\theta_k)\sin^l(\theta_k)\tan^m(\delta) a_k^q \tag{2.12}$$

2.2.2.2 Vehicle Throttle/Brake Model

The acceleration command obtained from the higher-level controller needs to be mapped to the throttle and the brake of the vehicle to control the velocity. This section will discuss the typical model of a powertrain system and its shortcomings. To get an accurate model of the powertrain, the engine and the transmission must be also modeled. However, to model the powertrain components, many variables need to be measured, such as the gear ratios in the transmission and the engine's power-torque curve. This downside makes it challenging to design a controller as the model parameters are typically unknown or hard to measure. The workaround to this problem is to design a speed controller that maps the desired velocity directly to the throttle and the brake. An example of a speed controller is the cruise control (CC). In modern vehicles, CCs are PI controllers that take the velocity error signal (reference velocity subtracted from the current velocity) and maps it to the pedals. The PI controller's major downside is that the model is unknown, thus making it more challenging to guarantee performance to a certain level. A novel approach was designed and experimented and yielded excellent results. The idea is to map acceleration directly to the pedals using ML techniques and a correction term to the model to account for any inaccuracies that external factors can cause, such as road conditions and gradient. Since there are no physics based models to use to extract features from, these are the intuitions considered when modeling the longitudinal controller:

- A vehicle has a maximum and a minimum acceleration,
- the acceleration of the vehicle is a function of the current velocity,

- the pitching and the rolling of the vehicle can be accounted for using the correction term,
- and the pedals control the acceleration and not the velocity of the vehicle.

Next was to come up with candidate functions that can explain the intuition. Logarithmic functions and linear functions are functions to be the best fit to map the pedals' acceleration. Therefore the following equation is the feature set for the throttle and brake:

$$Pedal_k = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{h=0}^{n}\sum_{l=0}^{n} W_{i,j,h,l} * v_k^i a_{k+1}^j \ln^h(v_k) \ln^l(a_{k+1}) \tag{2.13}$$

A third model will be designed to find the ideal time to switch between the throttle and the brake. This model will be called the coasting model. The coasting model would predict the acceleration, a_c, if the vehicle were to coast (no throttle or brake application). The intuition behind the coasting model is that there is a maximum deceleration. Therefore a logarithmic function was also fit to model the coasting behavior of the vehicle.

$$a_c = \sum_{i=0}^{n}\sum_{j=0}^{n} W_{i,j} * v_k^i \ln^j(v_k) \tag{2.14}$$

2.3 Predictors

There are two main classes of predictors; Linear and Nonlinear predictors. Given the assumptions of the engineer, one can use either to obtain powerful predictive models that are used in many different applications. Since MPCs require the model to be continuous and continuously differentiable, only linear predictors can be used as models. On the other hand, the longitudinal model does not carry any assumptions on the model, therefore nonlinear predictors were used.

2.3.1 Linear Predictors

Linear predictors are of the following form:

$$\hat{Y} = \vec{\alpha} X \tag{2.15}$$

where $\hat{Y}$ is the prediction, $\vec{\alpha}$ are the coefficients, and X is the feature set. The goal of the linear regression is to minimize the following objective function:

$$J = \sum_{i=0}^{n} (Y_i - \hat{Y}_i)^2 \tag{2.16}$$

where J is the cost function and Y_i are the measurements, There are many ways to solve for this problem, however, the most popular method is known as Gradient Descent. Its generally used for large datasets as its more memory efficient and can work well for non-convex optimization problems.

With that being said, the criteria for picking the linear predictors was it must include L_1 and L_2 terms, therefore for this thesis, LASSO and Elastic Net regression will be used for evaluations.

2.3.1.1 LASSO Regression

The objective function for LASSO regression:

$$J = \sum_{i=0}^{n} (Y_i - \hat{Y}_i)^2 + \lambda L_1 \tag{2.17}$$

From the equation, the solution of LASSO is the same solution of ordinary least squares when λ is equal to 0. Since LASSO includes the L_1 term, it helps with feature selection as well as improving over-fitting. One downside of LASSO is that it is a parameter based predictor, which means that you can get a different model every time you change λ.

2.3.1.2 Elastic Net Regression

The objective function for Elastic Net regression [18]:

$$J = \sum_{i=0}^{n} (Y_i - \hat{Y}_i)^2 + \lambda(\frac{1-\beta}{2} L_2 + \beta L_1) \tag{2.18}$$

Elastic Net regression has both L_1 and L_2 terms in the objective functions, with λ and β as tuning parameters. By tuning the parameter β, you can tune the model to make L_1 or L_2 to be more predominant.

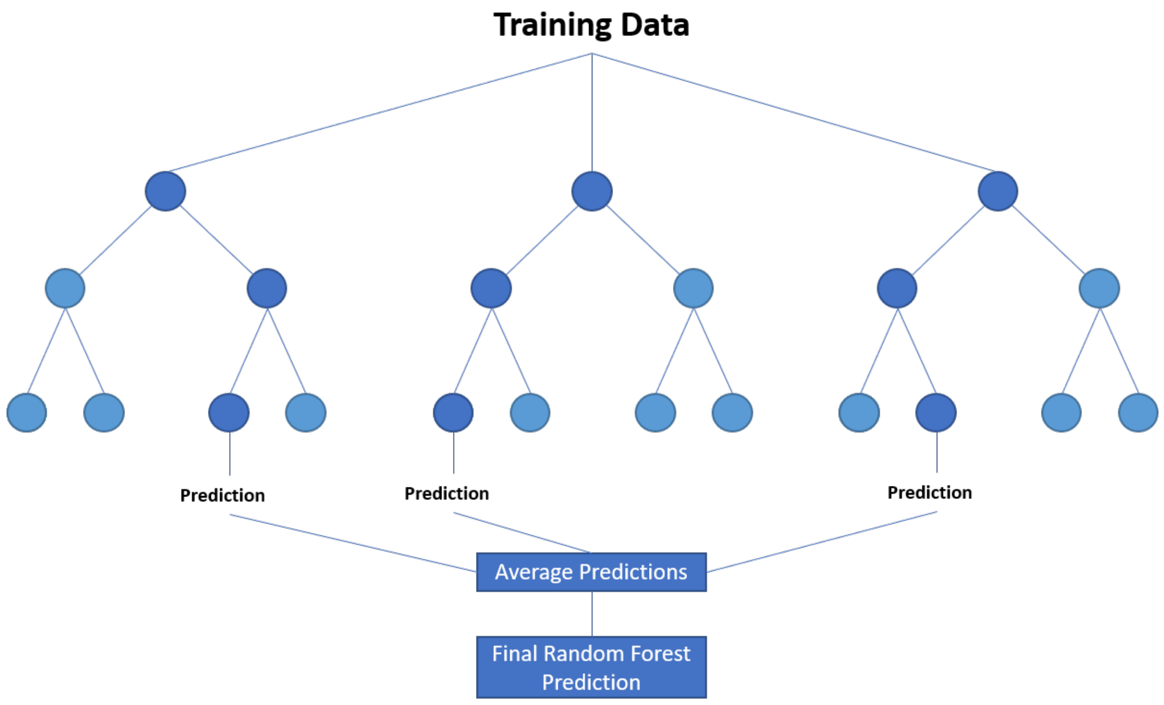


Figure 2.4: Diagram of Random Forest Algorithm.

2.3.2 Nonlinear Predictors

Nonlinear predictors can come in many forms such as Naive Bayes, Decision Trees, Support Vector Machines, etc. This thesis will compare two different types of decision trees, Random Forest and XGBoost, that use different techniques for training the model.

2.3.2.1 Random Forest

Random Forest (RF) is a meta-estimator, which means it uses many predictions from multiple decision trees to come up with one final prediction [19]. The technique that RF uses to generate the predictors is called bootstrap aggregation, or bagging for short. With bagging you form many decision trees in parallel, and each tree trains on a random sample of the training data. Once the decisions trees have been trained, the prediction of RF is the average prediction from all the decision trees.

Due to bagging, RF is a very powerful prediction model as it can handle noisy data by rejecting outliers and performs feature selection inherently by training multiple smaller decision trees. However, RF may show how important each of the features are, the model is still not interpretable and can be only treated as a black box model.

2.3.2.2 XGBoost

XGBoost (XGB) is similar to RF as it is also a meta-estimator [20]. However, XGB uses a different technique called boosting to generate the decision trees. With boosting, the trees are generated sequentially and the training samples are weighted according to how difficult it is to learn those specific data points. So for example, the first tree assumes that all training samples are equally hard to learn, and once its done training it tells the next tree which samples it had difficulties with, and so forth.

The advantages of XGB algorthim heavily optimized for computational efficiency which is important for safety critical systems, and like RF, XGB is less susceptible to over-fitting and outliers. The downside of XGB is that it is still considered a black box model and it has 7 tunable parameters that can effect the performance of the model.

2.3.3 Parameter Optimization

All the aforementioned predictors require heavy parameter tuning in-order to get a good performing model. There are naive approaches such as Grid Search and Random Search that can solve the tuning problem, however, these approaches also depend on parameters that the engineer has to manual select. The approach this thesis explored is called Bayesian Optimization [21]. Bayesian optimization iteratively randomly samples N number of parameters to evaluate at each iteration, and it trains a Bayesian model called the "Surrogate Model" to suggest which parameters to evaluate next to improve the predictors. Over time, the Bayesian has more accurate representation of the parameter space. By having a Surrogate Model predicting which parameters to evaluate next, you eliminate many parameter combinations from the parameter space that will deteriorate the performance of the predictors, thus reducing the number of iterations to find the best combinations of parameters.

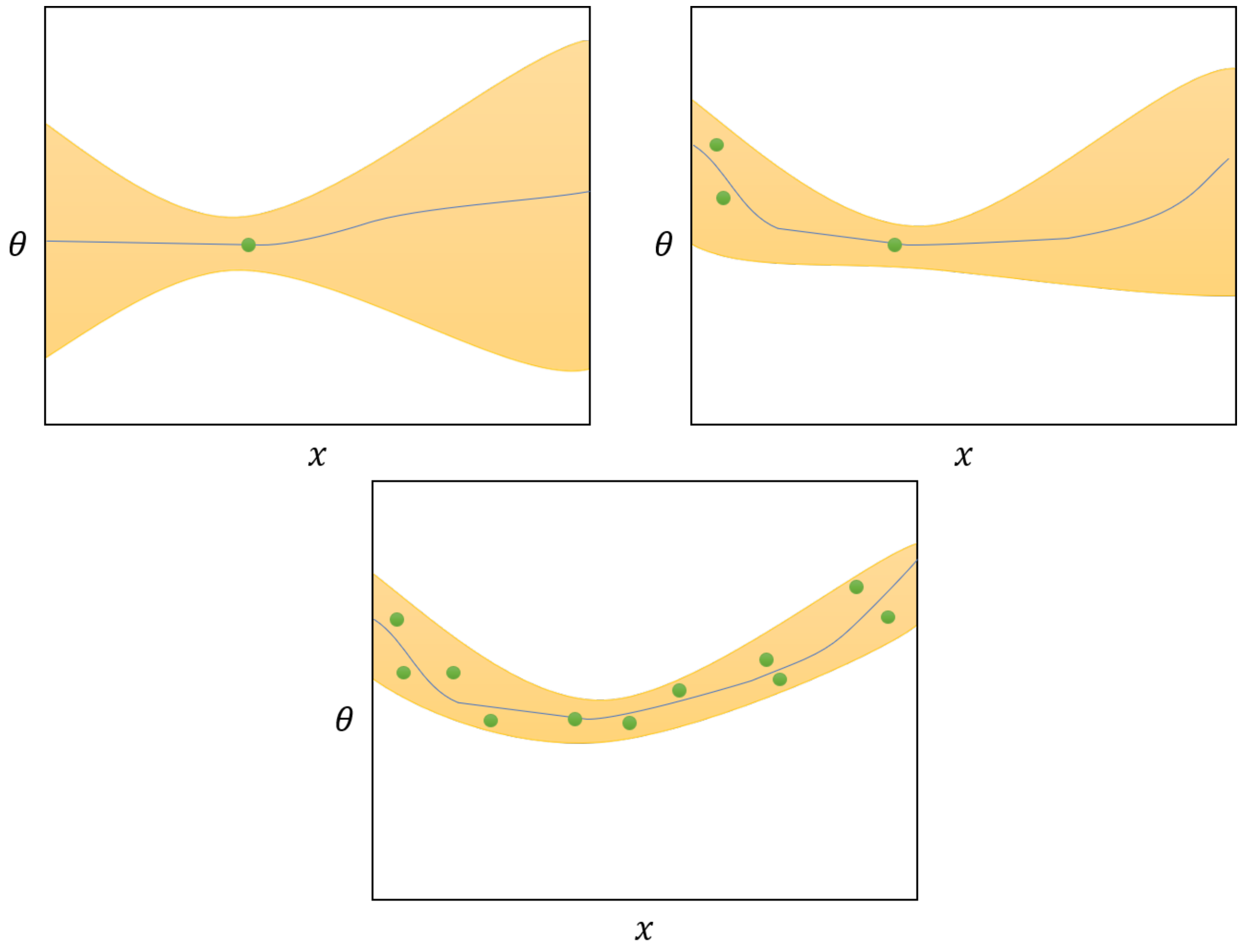


Figure 2.5: Bayesian Linear Regression diagram.

3. CONTROLLER DESIGN

The goal of the thesis is to design a lateral and a longitudinal controller for the Warthog and ASV. The controller must be robust to different road and vehicle conditions, takes into account the physical constraints of the system, and has to be computationally efficient. Nonlinear Model Predictive Controller (MPC) was selected for this thesis as it is a robust and optimal controller and considers state and input constraints. In this chapter, the concept of Optimal Control Problem (OCP) and MPC will be introduced, the different techniques to solve for the MPC will be discussed, and how the controllers were designed for the Warthog and ASVs.

3.1 Optimal Controller Formulation

OCPs are control law obtained from an optimization problems that takes into account the dynamics of the system, the constraints of the system, and the desired reference trajectory for the system [22]. There are three main motivations for OCPs, the first is to get the optimal control law given the conditions of the system, the second is to understand what might the limits of the system be, and lastly get a control law that is acceptable given some external constraints such as time limit or computational cost.

In each control design, the domain of the system (time or frequency domain) must be selected first. In this thesis, time domain optimal controller was explored since dynamics of the model was identified in the time domain. The optimization variables must be then selected. In most cases, the optimization variable would be the input of the system, however, the states of system can also be considered as optimization variables and will be discussed later in the chapter. The cost function and constraints must then be designed and tuned to get the desired behavior of the system. The cost function and constraints can be nonlinear, however, they must be continuously differentiable. The optimization problem is as follows:

$$\begin{aligned} \arg\min_{U} \quad & L(Z_k, Z_{ref}, U_k) \\ \text{s.t.} \quad & Z_{k+1} = f(Z_k, U_k) \\ & g_l \leq h(Z_k, U_k) \leq g_u \end{aligned} \tag{3.1}$$

Where Z_k is the current state of the system, Z_{ref} is the reference point or setpoint of the system, U_k is the input of the system, the function f is the dynamic model of the system, the function h are the constraints of the system with upper and lower bounds, g_l and g_u respectively.

3.1.1 Model Predictive Controllers

MPCs is considered a finite-time horizon OCP, in other words, an OCP that is solved N time-steps into the future. MPCs can handle nonlinear cost functions and constraints and the optimization can be solved in real-time. There are multiple techniques to solve MPCs [23], however, the optimization problem must be defined first:

$$\begin{aligned} \arg\min_{Z,U} \quad & \sum_{i=0}^{N} L(Z_k, Z_{ref}, U_k, U_{ref}) \\ \text{s.t.} \quad & Z_{k+1} = f(Z_k, U_k) \\ & g_l \leq h(Z_k, U_k) \leq g_u \end{aligned} \tag{3.2}$$

Where N is the horizon. The first technique is called Direct Single Shooting Method. Which is a sequential method, and it tries to optimize the input at each time step and propagates the states forward (Figure 3.1). The issue with single shooting is that you might find the optimal solution at each time step, but the whole trajectory might not be optimal. For nonlinear systems, the convergence to a solution is not guaranteed.

On the other hand, Direct Multiple Shooting Method alleviates the issues encountered by the single shooting method. Multiple shooting is a simultaneous method, the optimizer solves for the optimal solution for each time-step simultaneously. The optimization problem becomes bigger as the optimization variables are both the states and the inputs of the system. However, multiple shooting guarantees local optimality for the entire trajectory and convergence for nonlinear

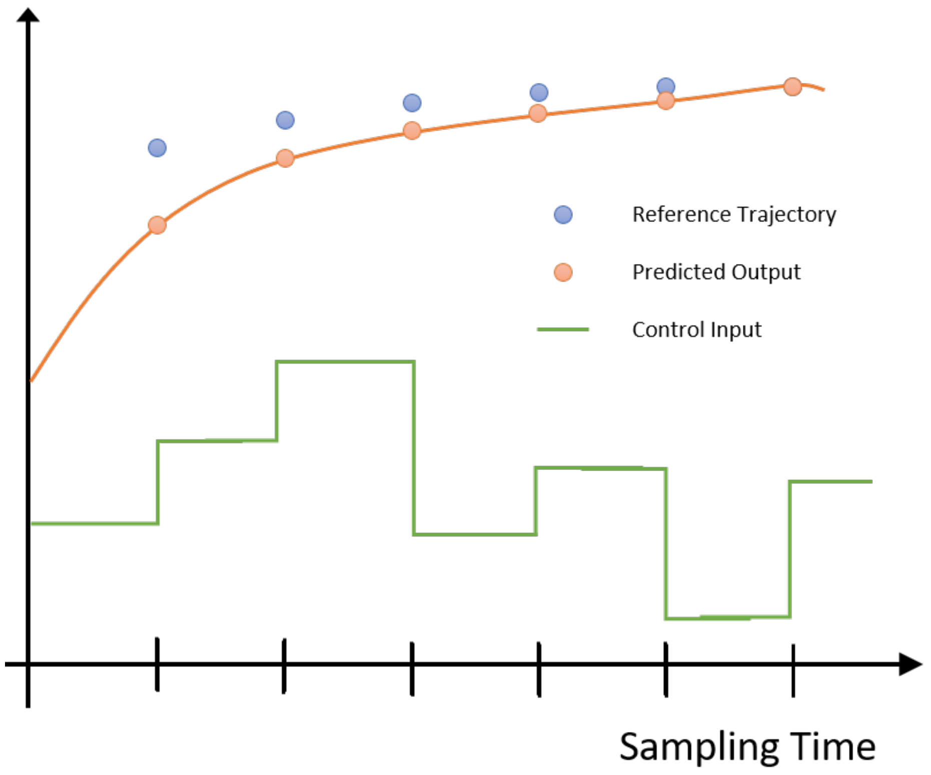


Figure 3.1: Diagram of the Direct Single Shooting Method.

systems.

3.1.2 MPC for Warthog

Before designing the controller for the Warthog, the constraints of the system should be studied first. Table 3.1 lists the physical constraints of the Warthog.

Table 3.1: Warthog parameters and constraints.

Parameter	Lower Limit	Upper Limit
Wheel Base (m)	0.91	N/A
Control Rate (Hz)	N/A	50
Maximum Lateral Acceleration ($\frac{m}{s^2}$)	-2	2
Maximum Longitudinal Acceleration ($\frac{m}{s^2}$)	-2	2
Maximum Tangential Speed ($\frac{m}{s}$)	-5	5
Maximum Tangential Acceleration ($\frac{m}{s^2}$)	-50	50
Maximum Yaw Rate ($\frac{rads}{s}$)	-4	4
Maximum Yaw Acceleration ($\frac{rads}{s^2}$)	-40	40

Even though the Warthog can handle higher lateral and longitudinal accelerations, they were limited to 2 m/s^2 to protect the robot from damages due to bumpy off-road terrains. The rest of the constraints were set by the manufactured of the Warthog, ClearPath Robotics.

The cost function was designed to minimize the cross-track error, which is the distance between the desired path and the current position of the robot (Equation 3.3), the velocity error (Equation 3.4), the heading error (Equation 3.5), and the input error (Equations 3.6 and 3.7).

$$\text{CTE}^2 = (x_k - x_{ref})^2 + (y_k - y_{ref})^2 \tag{3.3}$$

$$v_{error}^2 = (v_k - v_{ref})^2 \tag{3.4}$$

$$\theta_{error}^2 = (\theta_k - \theta_{ref})^2 \tag{3.5}$$

$$v_{r,error}^2 = (v_{r,k} - v_{r,ref})^2 \tag{3.6}$$

$$v_{l,error}^2 = (v_{l,k} - v_{l,ref})^2 \tag{3.7}$$

The errors had to be squared since square root functions are not continuously differentiable. Therefore, the cost function is as follows:

$$J_{warthog} =< \vec{C}, [\text{CTE}^2, v_{error}^2, \theta_{error}^2, v_{r,error}^2, v_{l,error}^2] > \tag{3.8}$$

Where $\vec{C}$ is the tunable coefficients vector for the objective function. The optimization problem for the Warthog's controller is showcased in Equation 3.9.

$$\begin{aligned}
\arg\min_{Z,U} \quad & \sum_{i=0}^{N} J_{warthog,i} \\
\text{s.t.} \quad & Z_{k+1} = f(Z_k, U_k) \\
& -2\frac{m}{s^2} \leq \dot{v}_r, \dot{v}_l \leq 2\frac{m}{s^2} \\
& -2\frac{m}{s^2} \leq \frac{v^2}{r} \leq 2\frac{m}{s^2} \\
& -5\frac{m}{s} \leq v \leq 5\frac{m}{s} \\
& -50\frac{m}{s} \leq \dot{v} \leq 50\frac{m}{s} \\
& -4\frac{rads}{s} \leq \omega \leq 4\frac{rads}{s} \\
& -40\frac{rads}{s^2} \leq \dot{\omega} \leq 40\frac{rads}{s^2}
\end{aligned} \tag{3.9}$$

3.1.3 MPC for Ackermann Steering Vehicles

For ASVs, the controller can be tuned for different scenarios; emergency maneuvers, city driving, highway driving, etc. For this thesis, the controller was designed for city and highway driving. No-slip was also assumed for the system, so a maximum lateral acceleration of 2.5 $\frac{m}{s^2}$ was imposed

[4, 24, 25]. The vehicle that is used for experimentation in simulation is the Tesla Model 3.

Table 3.2: Tesla Model 3 parameters and constraints.

Parameters	**Lower Limit**	**Upper Limit**
Wheel Base m	2.875	N/A
Control Rate Hz	N/A	15
Maximum Steering $Rads$	-1.22	1.22
Steering Ratio	N/A	15
Maximum Steering Wheel Speed $\frac{rads}{s}$	-8	8
Maximum Lateral Acceleration $\frac{m}{s^2}$	-2.5	2.5
Maximum Longitudinal Acceleration $\frac{m}{s^2}$	-2.5	2.5

The cost function was designed to minimize the cross-track error (Equation 3.3), the velocity error (Equation 3.4), the heading error (Equation 3.5), and the input error (Equations 3.10 and 3.11). However, during testing these factors were not enough to design a stable controller that can handle different road gradients and radii. Other cost functions were added to improve the controllers performance (Equations 3.12 - 3.16).

$$a_{error} = (a_k - a_{ref})^2 \tag{3.10}$$

$$\delta_{error} = (\delta_k - \delta_{ref})^2 \tag{3.11}$$

$$\text{CK}_{error} = \text{CK}_k v_k^2 \tag{3.12}$$

$$\text{CTE}_v = \text{CTE}_k^2 v_k^2 \tag{3.13}$$

$$\theta_v = \theta_{error}^2 v_k^2 \tag{3.14}$$

$$\dot{a}_{error} = \dot{a}_k (v_k^2 + 1) \tag{3.15}$$

$$\dot{\delta}_{error} = \dot{\delta}_k (v_k^2 + 1) \tag{3.16}$$

Where CK is the instantaneous curvature of the road. The goal behind CK$_{error}$ is to encourage the optimizer to lower the speed of the vehicle if the curvature is high. This might increase the velocity error, however, from testing it improved the cross-track error. Similar idea used with CTE$_v$ and θ_v; if the cross-track and heading angle errors are too high, then the optimizer is encouraged to lower the speed of the vehicle. The derivative of the acceleration (jerk) of the vehicle was also penalized. Penalizing the jerk of the vehicle allows for smoother application of the throttle and the break. The steering rate was also penalized for smoother steering applications. At higher speeds, having a high steering rate can be catastrophic and can lead the controller to be unstable. Therefore, the steering rate was multiplied by the velocity of the vehicle. Both inputs were multiplied by the velocity. The cost function for ASVs is as follows:

$$J_{ASV} =< \vec{C}, [\text{CTE}^2, v_{error}^2, \theta_{error}^2, a_{error}, \delta_{error}, \text{CK}_{error}, \text{CTE}_v, \theta_v, \dot{a}_{error}, \dot{\delta}_{error}] > \tag{3.17}$$

The optimization problem for the Warthog's controller is showcased in Equation 3.18.

$$\begin{aligned}
\arg\min_{Z,U} \quad & \sum_{i=0}^{N} J_{ASV,i} \\
\text{s.t.} \quad & Z_{k+1} = f(Z_k, U_k) \\
& -8.0\frac{rads}{s} \leq \dot{\delta} * \text{steering ratio} \leq 8.0\frac{rads}{s} \\
& -2.5\frac{m}{s^2} \leq \frac{v^2 \tan(\delta)}{L} \leq 2.5\frac{m}{s^2} \\
& -2.5\frac{m}{s^2} \leq v^2 \leq 2.5\frac{m}{s^2} \\
& -1.22rads \leq \delta * \text{steering ratio} \leq 1.22rads
\end{aligned} \tag{3.18}$$

3.2 Longitudinal Controller

The higher-level controller described in Equation 3.18 only outputs the desired acceleration and steering angle for the vehicle. However, to control the acceleration of the vehicle, the higher-level acceleration command must be mapped to the throttle or brakes of the vehicle. The naive approach is to directly use the model obtained in Equations 2.13 and 2.14. The downside of only using these models to control the system is that these models assume a flat road surface when in fact road gradients are almost never flat, especially for highway entrances and exits. The idea is to add a correction term to these models that takes into account the gradient changes (Figure 3.2). This work is inspired by the work done in Unmanned Systems Lab at TAMU, where PI controllers

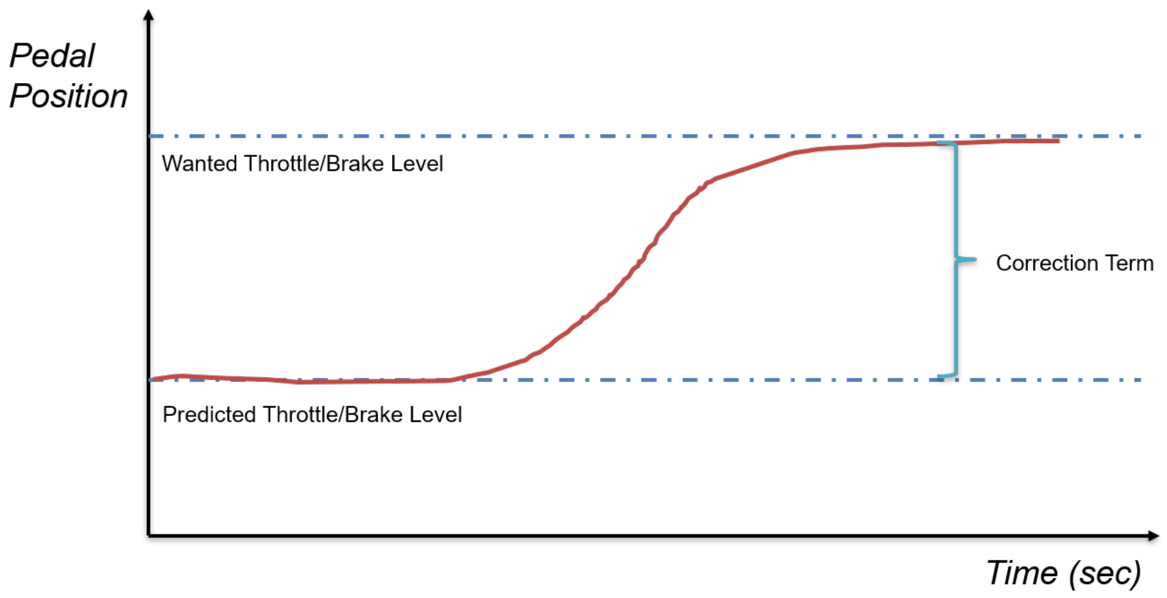


Figure 3.2: Diagram to showcase the correction term.

were used to control the throttle and brakes and added a feed forward term to correct the controllers (Figure 3.3). The PI controllers use the velocity error as the signal.

However, PID controllers fail to stabilize the vehicle in downhill and uphill scenarios. Assume a vehicle's speed is at reference speed on a downhill, therefore the PID should output 0. Which is counter-productive. The required output for the correction term should be a negative constant value to slow the vehicle down. The vehicle would then exceed the reference velocity and only then the PID would output a negative value. Overtime, PID would keep fluctuating towards 0 and averaging a negative value, thus never stabilizing the system. This thesis proposes to use the summation of

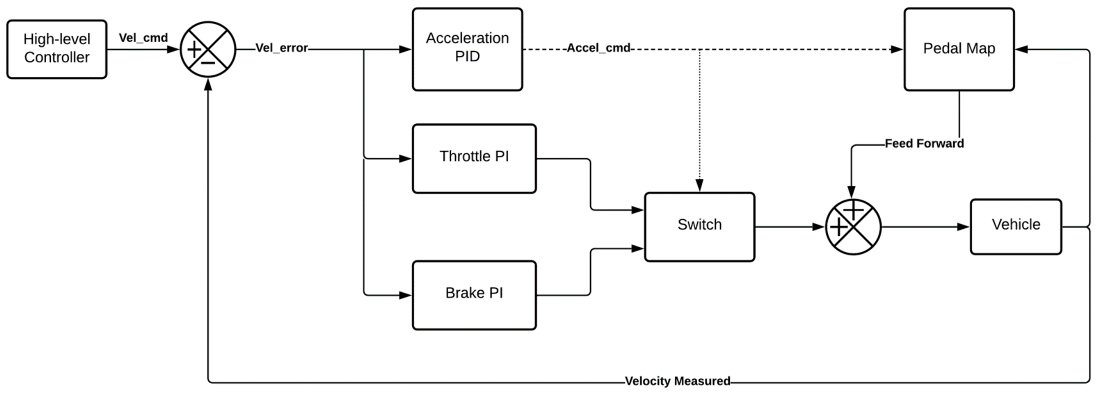


Figure 3.3: Longitudinal controller developed by the Unmanned Systems Lab.

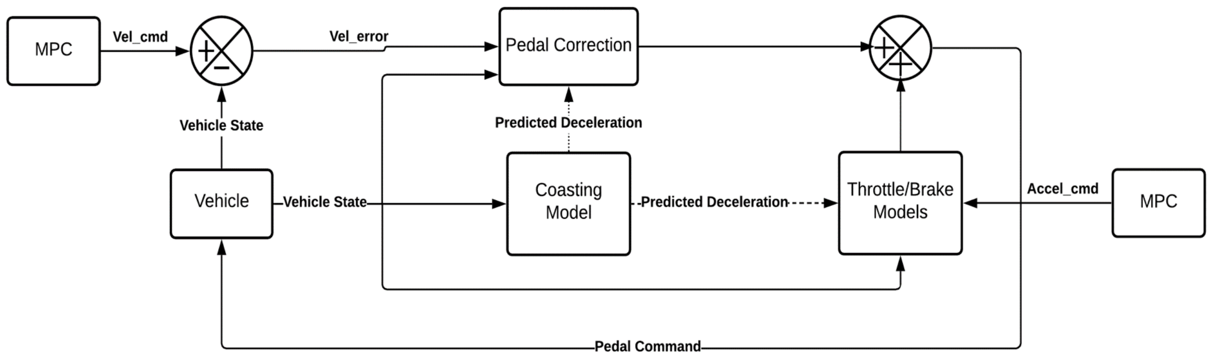


Figure 3.4: Proposed Data-Driven longitudinal controller.

previous PID outputs as the correction term instead. By summing the PID outputs, it eliminates the fluctuation phenomena experienced by the PID and would output a constant negative value over a short period of time, thus stabilizing the vehicle. The downside of this approach, however, is the controller will always be over damped, regardless of how it is tuned. Figure 3.4 and Equation 3.19 showcase the new proposed longitudinal controller.

$$\text{Longitudinal Control Law} = \text{Pedal}_k + \sum_{i=-\infty}^{0} \text{PID}(v_{error}) \tag{3.19}$$

4. RESULTS AND DISCUSSION

This thesis validated results using simulation for both the Warthog and the Tesla Model 3. The simulation setup is first introduced and discusses the tools developed to emulate the behavior of the actual systems. Results for system identification for the Warthog and Tesla Model 3 is then discussed. Different testing scenarios were generated to evaluate the performances of the MPCs.

4.1 Experimental Set-Up

4.1.1 Warthog Interface and Data Collection

The sensor stack on the Warthog allows for accurate position and velocity estimate in an off-road environment. The Warthog is fitted with LIDARS (Ouster and Velodyne) and GNSS (Vectornav) for local planning (Figure 4.1). Local planning allows for the robot to navigate safely in unstructured environment, and even in GPS denied regions. However, the raw information gathered from the sensor stack is too noisy to reliably control the Warthog. With that being said, using the data obtained from the sensor stack, an on-board computer deploys an Extended Kalman Filter algorithms to estimate the position and velocity of the robot. The on-board also translates high-level control commands, known as Twist commands, to low-level control; it converts tangential velocity (v) and yaw rate (ω) commands to wheel speeds through a proprietary driver.

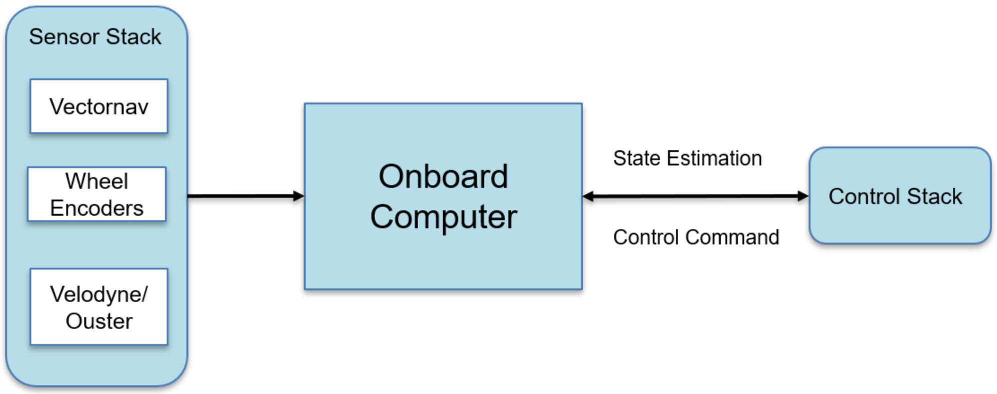


Figure 4.1: Warthog's Interface with the on-board computer and the sensor stack.

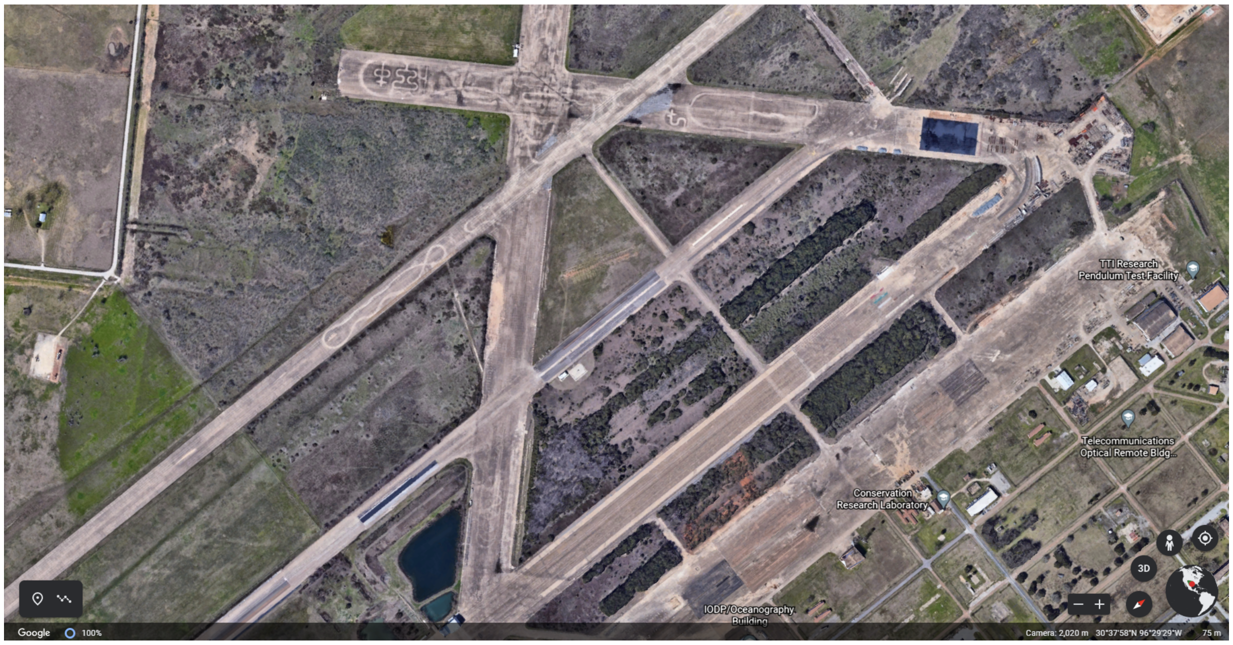

TTI Research Pendulum Test Facility
Telecommunications Optical Remote Bldg...
Conservation Research Laboratory
IODP/Oceanography Building
Google
Camera: 2,020 m 30°37'58"N 96°29'29"W 75 m

Figure 4.2: Google Earth image of RELLIS Campus.

For data collection, a remote control was used to drive around the robot. The testing was conducted on RELLIS campus, seen in Figure 4.2. It is very crucial to collect a diverse set of datasets in-order to capture as much of Warthog's dynamics as possible. However, it is practically impossible to quantify how diverse a dataset is. Therefore, while collecting data, qualitative observations were made about the environment to avoid any extrapolation of data. The dataset that was collected was on a sunny day, part of the data collected on a slightly damped bumpy grass terrain, the other part was on flat asphalt surface, and the Warthog was 50% charged. Therefore, any results obtained from system identification can be applied to these conditions only.

4.1.2 Simulation

Having a well-designed simulator in the arsenal of roboticists is a major contributor to the success of their research. It allows them to quickly iterate and test algorithms it cost effective manner. This thesis looked at two different simulators to test and evaluate the system identification and controller algorithms, Gazebo [26] and CARLA [27].

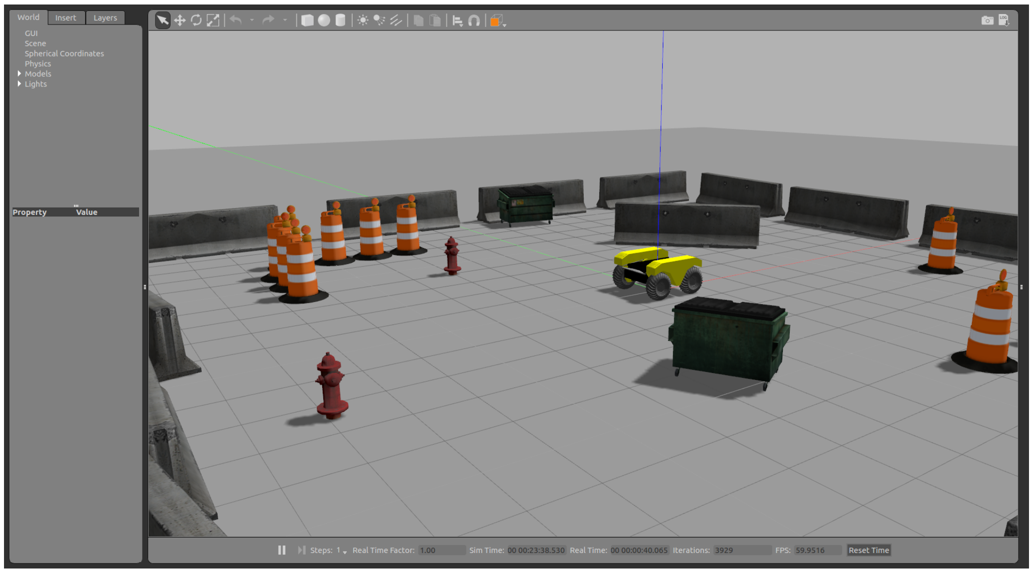


Figure 4.3: Warthog in the Gazebo simulation.

4.1.2.1 Gazebo

Gazebo is an open source 3D dynamic simulator that is widely used in the robotics community. Gazebo's physics engines accurately simulate multi-body robots in complex environments. Gazebo also offers simulations for a suit of sensors such as cameras and LIDARS. The two main uses for Gazebo are testing robotics software and designing robots. In this thesis, Gazebo will evaluate the performance of the Warthog. Luckily, ClearPath Robotics offers a Warthog simulation environment as well as simulating on the on-board computer (Figure 4.3). Due to the high-fidelity physics engine of Gazebo, testing the controller can give great insights on how the controller might behave on the actual system.

4.1.2.2 CARLA

CARLA is an open source simulation platform used to evaluate autonomous vehicles. CARLA is currently the standard as it is backed by industry heavy-weights such as Toyota and Mercedes. It provides digital assets such urban layouts and vehicles to construct the environment. CARLA can simulate traffic scenarios, dynamic weather, vehicle dynamics, and sensor suites. Unlike the Gazebo simulation where the Warthog simulation was readily available, more work has to be done

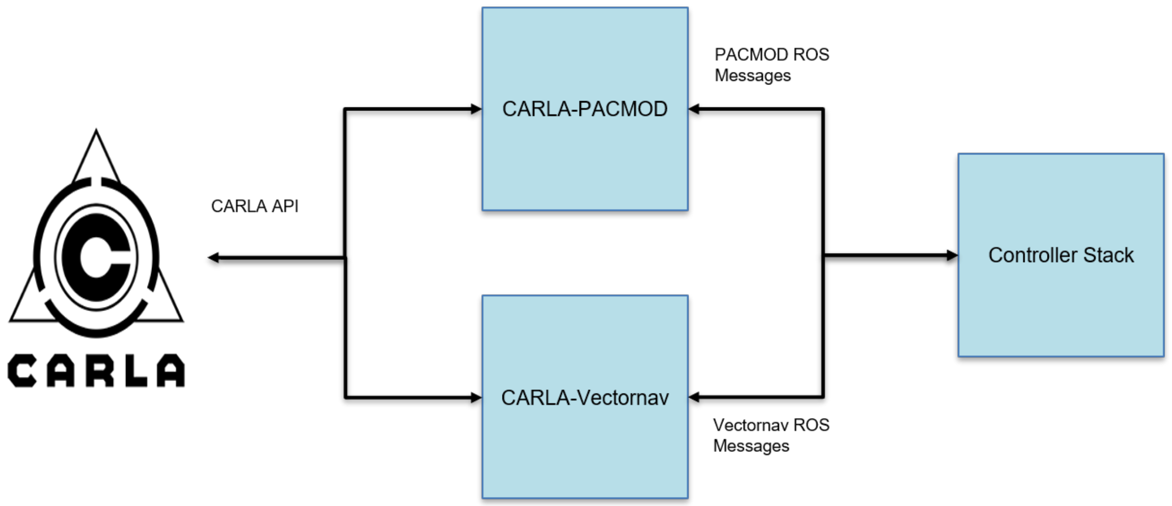


Figure 4.4: CARLA's simulation setup to communicate with the controller stack.

for CARLA. CARLA uses a server-client architecture, where the simulation lies on a server and the client communicates with the server using the provided Python API. The Python API allows you to control every aspect of the simulation, from vehicle controls to traffic lights in the city.

The vehicle's in Unmanned Systems Lab are fitted with PACMOD from AutonomousStuff, which acts as the middle-man between the actuation and the controller stack. The vehicles are also fitted with Vectornav for GNSS for positioning of the vehicle. The way each of the modules communicate between each other is using ROS, Robotic Operating System. To emulate the behavior of this setup, tools were developed to translate CARLA API to PACMOD and Vectornav, Figure 4.4.

CARLA provides many vehicles to experiment with. For this thesis, Tesla Model 3 was selected for experimentation. The data was collected by driving around "Town03" on flat roads using the Logitech F310 controller; the roads were dry.

4.2 System Identification Evaluation

This section will evaluate the performance of the proposed system identification technique in Chapter 2 for the Warthog and the Tesla Model 3. For the system identification, a second-order polynomial was used. The goal of this section is to figure out whether a data-driven approach to system identification outperform a physics-based model and how much data do you need in order to do so. The machine learning algorithms were solved using a python package called SciKit-Learn [28] on an Intel i7-8700k @ 3.70 Ghz.

4.2.1 Warthog System Identification

The Warthog was driven around RELLIS on off-road terrain as well as asphalt. It was driven in random maneuvers in-order to get the most diversified dataset to capture as much of the dynamics as possible. 17,155 data samples were collected at 50 Hz. Table 4.1 and 4.2 show the results for Elastic Net and LASSO for the position X model for the warthog.

Table 4.1: Elastic Net Regression performance for the Position X model for the Warthog.

Training Sample Size	1715	5146	8578	12009
Mean Error (m)	6.17E-05	5.34E-05	5.88E-05	6.332E-05
Standard Deviation (m)	0.0077	0.0073	0.0075	0.0077
% Improvement	-1.24	-0.33	0.12	1.33

Table 4.2: LASSO Regression performance for the Position X model for the Warthog.

Training Sample Size	1715	5146	8578	12009
Mean Error (m)	6.04E-05	5.33E-05	5.70E-05	6.04E-05
Standard Deviation (m)	0.00777	0.00730	0.00755	0.00777
% Improvement	-1.18	-0.32	1.27	1.56

Comparing LASSO to Elastic Net, LASSO outperformed Elastic Net regardless for all training sample size. This suggests that L_2 regularization has no effect on the performance of the model. Another interesting observation is that the standard deviation for both models are very similar. After determining that LASSO outperforms Elastic Net, does the model (Equation 4.1) make any physical sense?

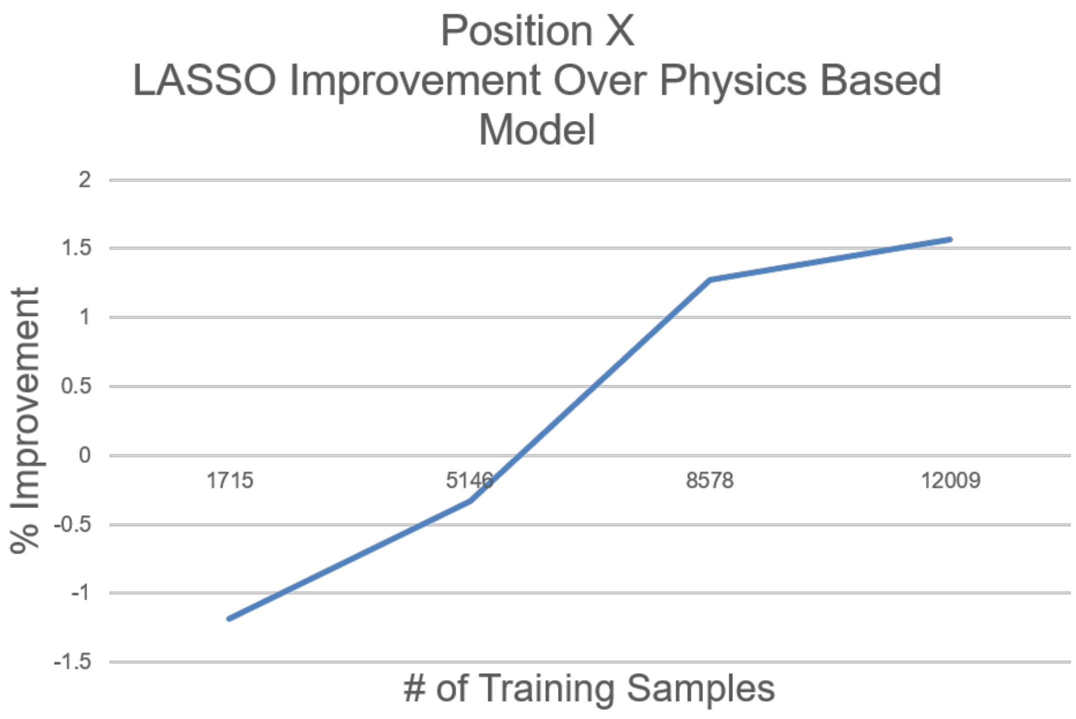


Figure 4.5: Evaluation of LASSO with respect to the physics-based model.

$$x_{k+1} = x_k + 0.02001 * v_k cos(\theta_k) \tag{4.1}$$

Even though the feature engineering step provided second-order nonlinear feature set, LASSO still returned the physics-based model with the parameters identified. The parameter that LASSO identified is Δt, which the nominal value is defined as 0.02 sec. The larger Δt value found through LASSO suggests that there might be a slight delay that hasn't been accounted for by the physics-based model. The next question to answer is did it improve over the physics-based model?

From Figure 4.5, there is a clear evidence that LASSO outperforms the physics-based model after training on 5146 samples, or 103 seconds worth of data. Keep in mind that these results can only explain the dynamics of the Warthog at the condition when the data was collected.

Table 4.3 and 4.4 show the results for Elastic Net and LASSO for the position Y model for the warthog.

Table 4.3: Elastic Net Regression performance for the position Y model for the Warthog.

Training Sample Size	1715	5146	8578	12009
Mean Error (m)	5.797E-05	5.539E-05	5.607E-05	5.595E-05
Standard Deviation (m)	0.007610	0.007429	0.007481	0.007459
% Improvement	-2.66	-1.37	1.41	2.82

Table 4.4: LASSO Regression performance for the position Y model for the Warthog.

Training Sample Size	1715	5146	8578	12009
Mean Error (m)	7.56E-05	5.51E-05	5.59E-05	5.53E-05
Standard Deviation (m)	0.007610	0.007429	0.007481	0.007441
% Improvement	-2.58	-1.23	1.61	2.98

Comparing the results, LASSO outperformed Elastic Net for all training sample size. For the position Y model, L_2 regularization hurt the performance of the predictor. Similar to the results to position X model, the standard deviation for both models are similar.

$$y_{k+1} = y_k + 0.02002 * v_k sin(\theta_k) \tag{4.2}$$

Equation 4.2 also makes physical sense; the output of the LASSO regression was the physics-based model with the parameters identified. This time, LASSO suggests that Δt to be 0.02002 seconds. Similar conclusion can be made, the physics-based model does not account for the delay in the system. Figure 4.6 shows that training the model with more than 5146 samples, the LASSO model is expected to outperform the physics-based model.

The last model that will be evaluated is the heading angle model. Table 4.5 and 4.6 show the results for Elastic Net and LASSO for the heading angle model for the warthog. Interestingly, both Elastic Net and LASSO yielded the exact same model for the heading angle, reaffirming that L_2 regularization is hurting the performance for the predictors. It also suggests that the Bayesian optimization weighted L_2 to be 0 for the Elastic Net model.

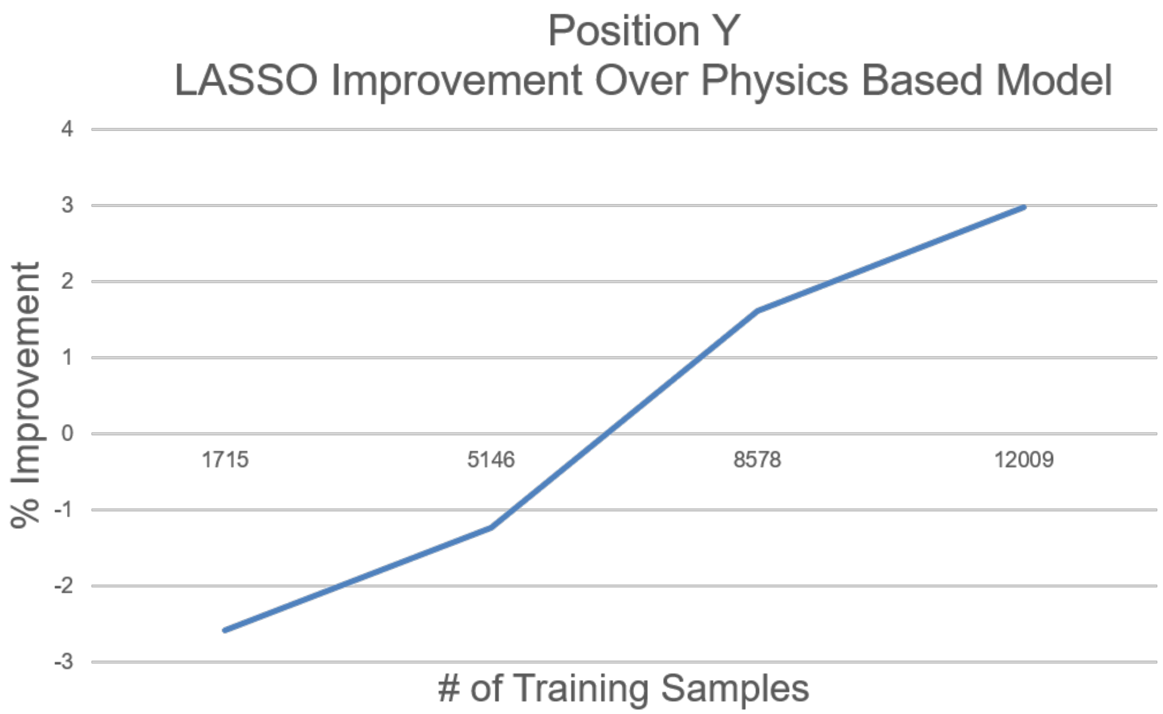


Figure 4.6: Evaluation of LASSO Position Y Model with respect to the physics-based model.

Table 4.5: Elastic Regression performance for the heading angle model for the Warthog.

Training Sample Size	1715	5146	8578	12009
Mean Error (rads)	8.976E-06	8.84E-06	9.321E-06	1.12E-05
Standard Deviation (rads)	0.00299	0.00297	0.00305	0.00335
% Improvement	-58.53	-58.97	-50.43	-39.54

$$\theta_{k+1} = \theta_k + 0.02002 * v_k sin(\theta_k) \tag{4.3}$$

The equation obtained for the heading angle (Eq. 4.3) also suggests that there might be a slight delay in the system. This time, the data-driven approach did not improve the physics-based model. Figure 4.7 shows that the LASSO model improved the physics-based model by -39.5%, which is a very large error. This can be attributed to the assumptions that were made about the Warthog, mainly the no-slip condition.

Table 4.6: LASSO Regression performance for the heading angle model model for the Warthog.

Training Sample Size	1715	5146	8578	12009
Mean Error (rads)	8.976E-06	8.84E-06	9.321E-06	1.12E-05
Standard Deviation (rads)	0.00299	0.00297	0.00305	0.00335
% Improvement	-58.53	-58.97	-50.43	-39.54

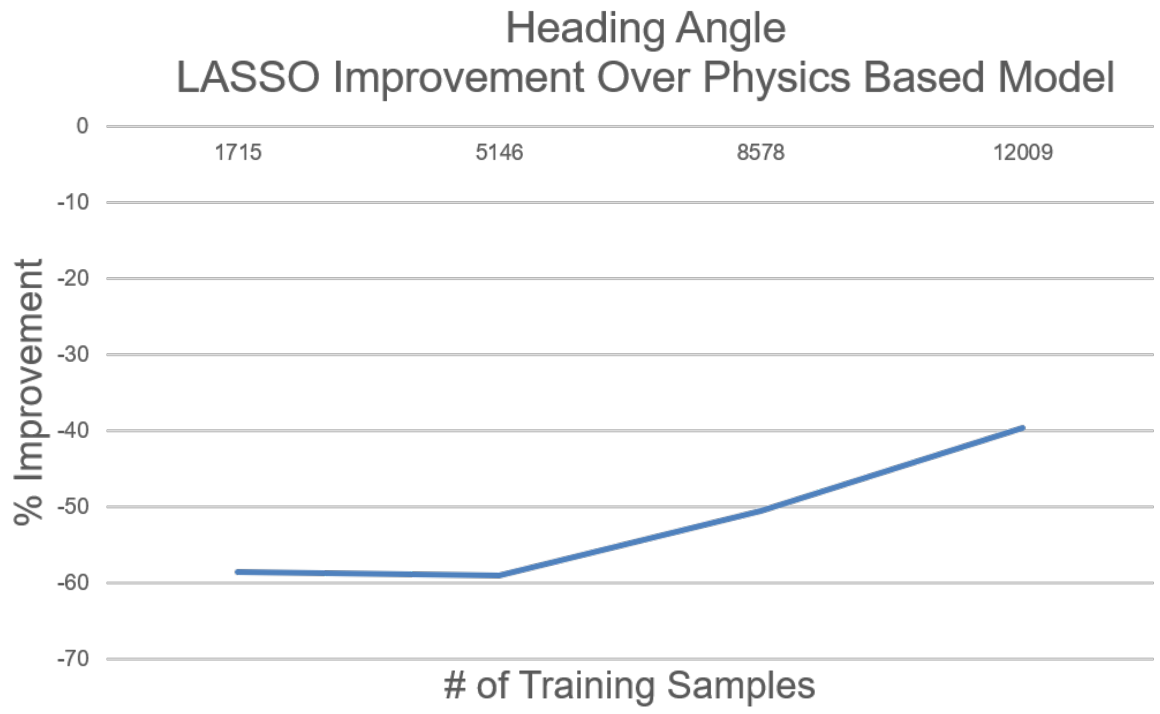


Figure 4.7: Evaluation of LASSO Heading Angle Model with respect to the physics-based model.

4.2.2 ASV System Identification

The Tesla Model 3 was driven around CARLA using a remote control at the physical limits of the vehicle. Since the data collection is done in simulation, the states and inputs of the vehicle can be assumed to be nominal with no uncertainty. 20,160 sample points were collected at 15 Hz; 70% of the collected samples were used for training. Table 4.7 and Table 4.8 show the results for Elastic Net and LASSO for the Tesla, respectively.

Both Elastic Net and LASSO regressions yielded similar results for the position X model and

Table 4.7: Elastic Net Regression performance for the Tesla Model 3.

	MSE	**Standard Deviation (m)**
X Model (m)	0.0452	0.214
Y Model (m)	0.0588	0.242
Yaw Model (rads)	6.00E-05	7.7E-03

Table 4.8: LASSO Regression performance for the Tesla Model 3.

	MSE	**Standard Deviation (m)**
X Model (m)	0.0459	0.214
Y Model (m)	0.0588	0.242
Yaw Model (rads)	6.00E-05	7.74E-03

exactly the same model for the position Y and the heading angle model. This finding confirms the findings found with the Warthog; L_2 regularization does not improve the prediction models for dynamic models. Equations 4.4 - 4.6 are the best models found after fitting them.

$$EN : x_{k+1} = x_k + 0.0669v\cos(\theta) \tag{4.4}$$

$$LASSO : y_{k+1} = y_k + 0.0668v\sin(\theta) \tag{4.5}$$

$$LASSO : \theta_{k+1} = \theta_k + 0.0669v\cos(\theta) \tag{4.6}$$

The position X and the position Y models improved over the physics-based model by 0.018% and 0.339%, respectively. This improvement is negligible, which suggests that the kinematic bicycle model is a good model to estimate the position of the vehicle, assuming it follows the aforementioned assumptions about the system. On the other hand, the heading angle model was improved by 11.11%. This can be attributed to the fact that the steering angle is assumed to be the average wheel angle of the two front tires.

Next is to evaluate the longitudinal controller. There is 18,350 samples collected for the throttle, 2,970 samples collected for the brake, and 3,080 samples collected for the coasting model.

Tables 4.9 and 4.10 show the results for the Random Forest and XGBoost longitudinal models, respectively.

Table 4.9: Random Forest Regression longitudinal models performance for the Tesla Model 3.

	MSE	**Standard Deviation (m)**
Throttle Model	0.011	0.104
Brake Model	0.058	0.241
Coasting Model ($\frac{m}{s^2}$)	2.30	1.51

Table 4.10: XGBoost Regression longitudinal models performance for the Tesla Model 3.

	MSE	**Standard Deviation (m)**
Throttle Model	0.0106	0.103
Brake Model	0.057	0.237
Coasting Model (m/s^2)	1.56	1.247

The throttle and brake values are between 0 and 1. XGBoost slightly edged over Random Forest for the throttle and the brake model, but it completely outperformed for the coasting model.

4.3 Controllers Evaluation

This section will evaluate the nonlinear MPCs performance for the Warthog in Gazebo and Tesla Model 3 in CARLA. The Warthog simulation environment was simple, therefore the goal is to test whether nonlinear MPC is robust and stable at physical limits of the robot. For the Tesla Model 3, three unique scenarios were setup to test specific aspects of the controllers.

The MPC was solved using the CASADI [29] framework with IPOPT [30] as the backend on an Intel i7-8700K @ 3.70 Ghz. The time-horizon for the Warthog was selected to 20 time-steps and the Tesla was 30 time-steps. The execution speed for both controllers was at 60 hz.

4.3.1 Warthog MPC Performance

To tune the Warthog's controller, trial and error was used. The following coefficients for the objective function was set to be:

$$\vec{C} = [64.0, 1.0, 0.5, 1.0, 1.0] \tag{4.7}$$

Increasing the weighting for CTE would force the robot to head towards the desired path but would also make the robot over-reactive and unstable. Increasing the weighting for θ_{error} would stabilize the system but would overdamp any cross-track error corrections. These two cost functions were the most important factors to tune as the dictated the behavior of the Warthog. The Warthog was then tested in an open, flat area in Gazebo. Figures 4.8 and 4.9 showcases the Warthog tracking a predefined path and the cross-track error over time, respectively.

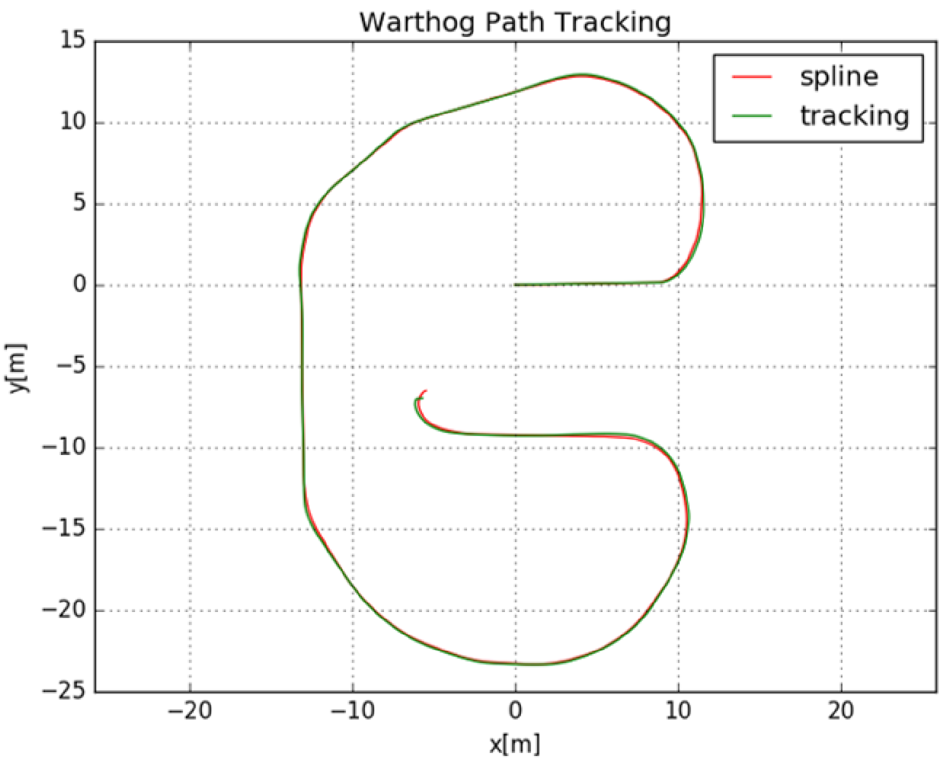


Figure 4.8: Warthog tracking a predefined path in Gazebo.

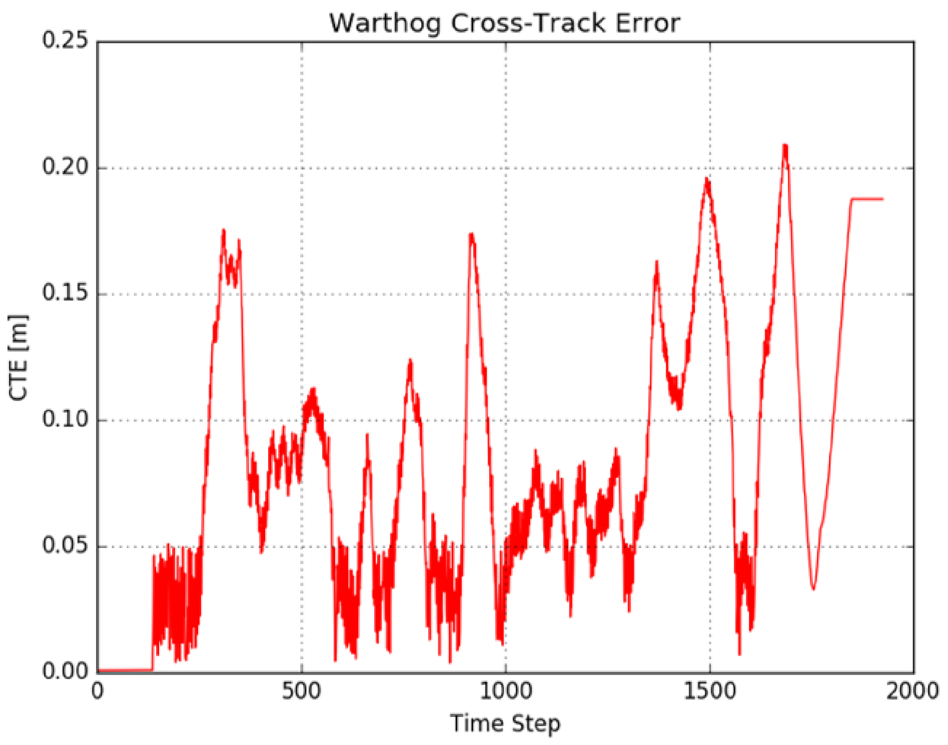


Figure 4.9: Cross-track error of the Warthog over time in Gazebo.

The MPC had an average CTE of 0.0813 m with a standard deviation of 0.055 m. The main challenge the controller faced is during corners; the Gazebo simulator ground was had lower traction than expected, which explains why it would veer of during corners. However, the controller is stable as it always forces the robot back in path without overshooting. Figure 4.10 shows the velocity tracking performance of the controller. The average velocity error was 0.315 $\frac{m}{s}$ with a standard deviation of 0.275 $\frac{m}{s}$.

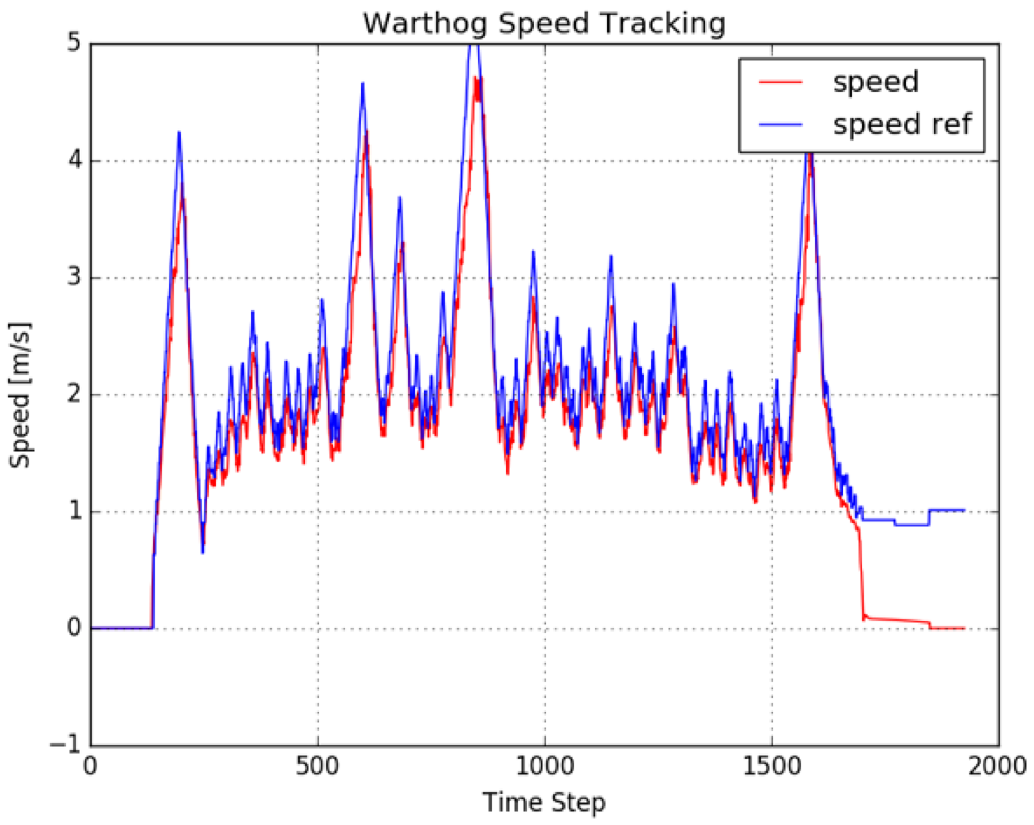


Figure 4.10: Warthog's MPC velocity tracking predefined reference velocities in Gazebo.

4.3.2 Ackermann Steering Vehicle MPC Performance

Tuning the Tesla's controller was also based on trial and error. The following coefficients were used for the objective function:

$$\vec{C} = [10.0, 800.0, 100.0, 3.0, 100.0, 200.0, 0.1, 20.0, 80.0, 80.0] \tag{4.8}$$

The behavior of the tuning of the controller was similar to the Warthog's; CTE causes the robot to over-react and the θ_{error} overdamped the system. However, the most important cost function was the v_{error}. If the velocity of the vehicle is slightly off form the desired trajectory, then the vehicle will overshoot the next turn and becomes unstable. It can be remedied by increasing the weight of θ_{error}, but this will force the vehicle to understeer in all turns.

4.3.2.1 Highway Exits/Entrances

This scenario was designed to test the ability of the controller to handle road gradient change for both the lateral and the longitudinal controller. The scenario consists of 1 km of highway exits and entrances. Figures 4.11 and 4.12 shows the tracking performance of Tesla's lateral controller and the cross-track error over time, respectively.

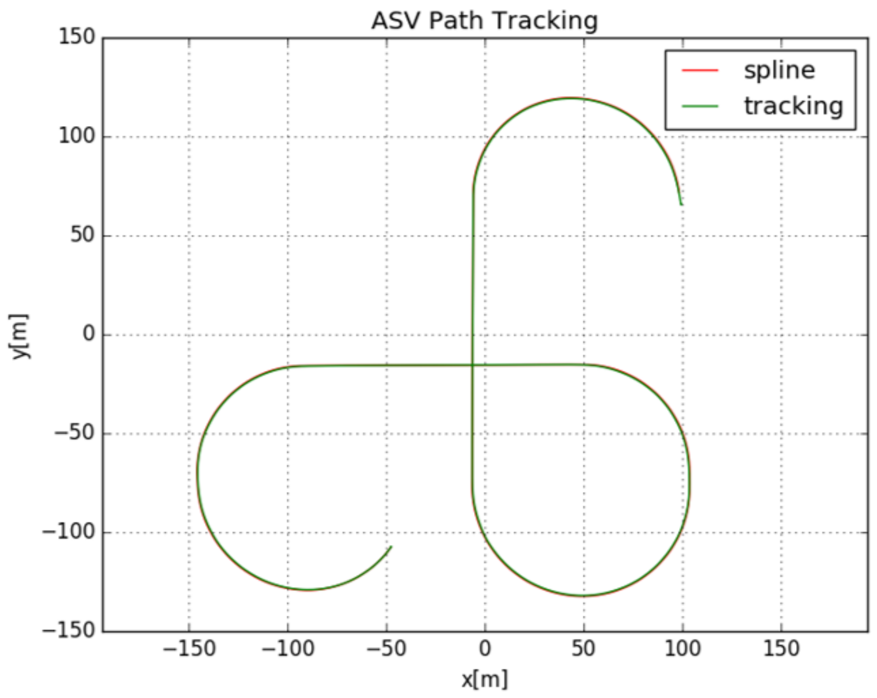


Figure 4.11: Tracking performance of Tesla with the Nonlinear MPC.

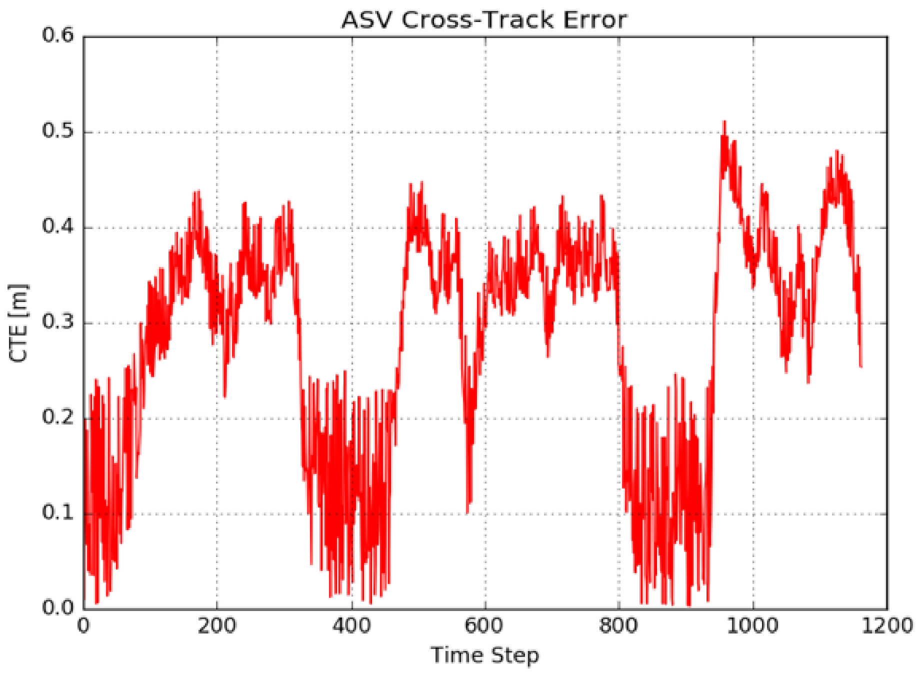


Figure 4.12: Cross-track error of Tesla with the Nonlinear MPC.

The controller had a cross-track error average of 0.28 m and a standard deviation of 0.12 m. The controller was able to control the vehicle during gradient change, even though the models assume a flat road. More importantly, the vehicle was able to track the velocity reference during the downhills and uphills of the the highway exits. Figure 4.13 showcases the longitudinal controller with and without the correction term. During the downhill segment of the exit, without adding the correction term, the vehicle overshot the reference velocity. Similar observation can be made with the uphill segment, without adding the correction term, the vehicle undershot the reference velocity.

The average velocity error is 0.33 $\frac{m}{s}$ with a standard deviation of 0.344 $\frac{m}{s}$. Figure 4.14 illustrates the heading angle tracking for highway exits. Having an accurate longitudinal model allows for a smoother application of throttle and brakes, which can be seen in Figure 4.15.

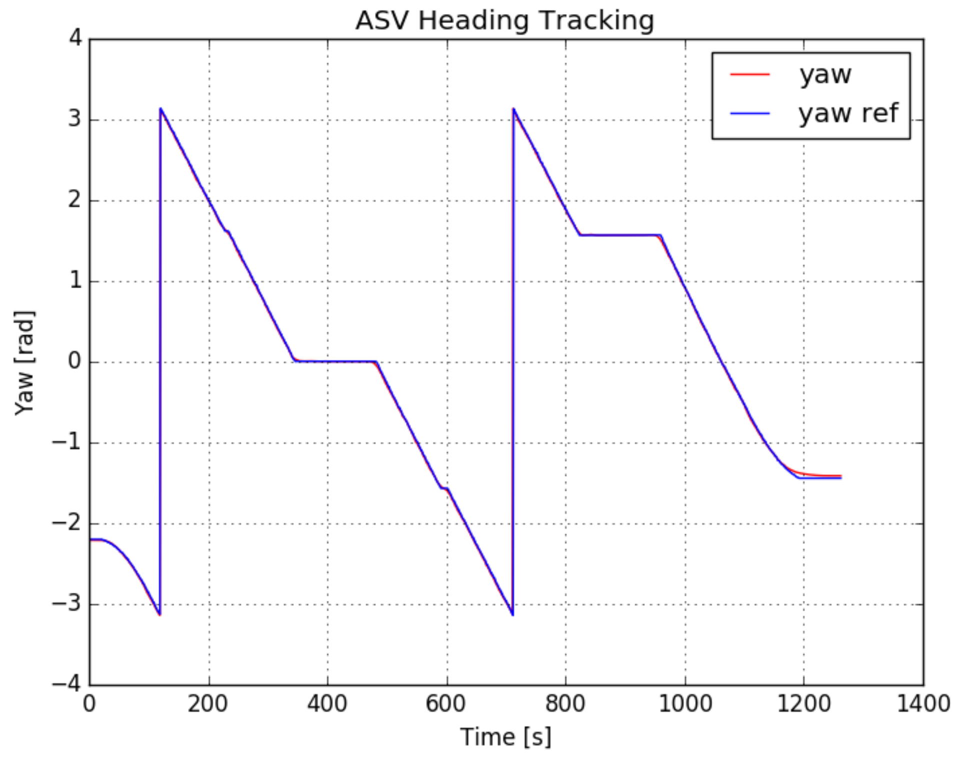
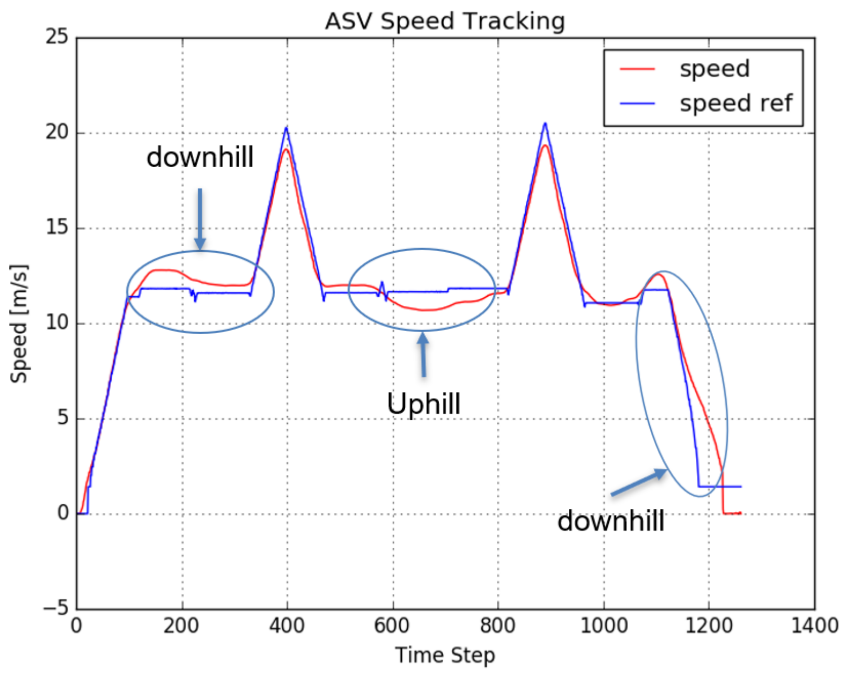
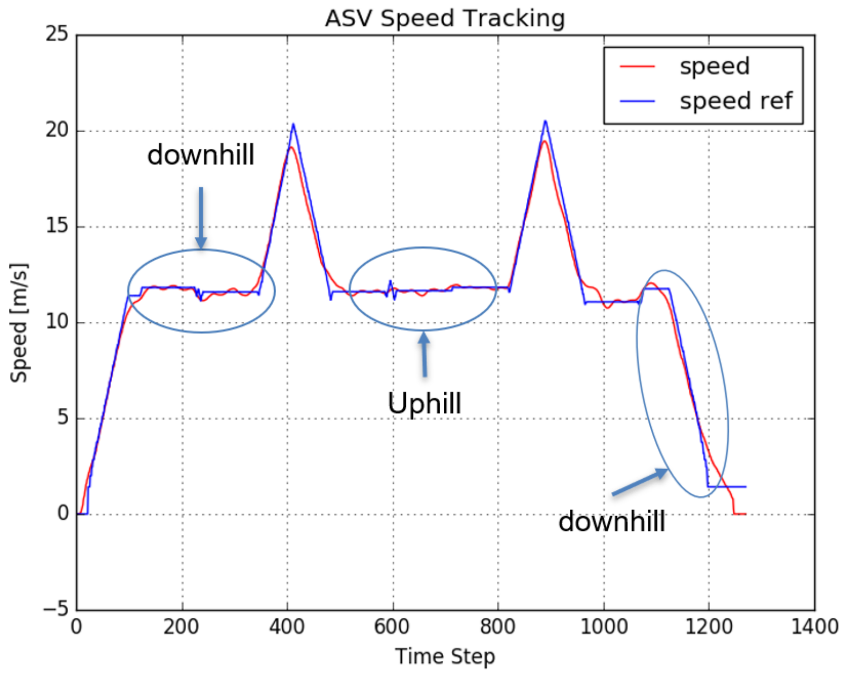


Figure 4.14: Yaw tracking performance of Tesla with the Nonlinear MPC for highway exits.



(a) Longitudinal Controller = Prediction Model



(b) Longitudinal Controller = Prediction Model + ΣPID

Figure 4.13: Velocity tracking of the longitudinal controller for highway exits. (a) Longitudinal controller without the correction term. (b) Longitudinal controller with the correction term.

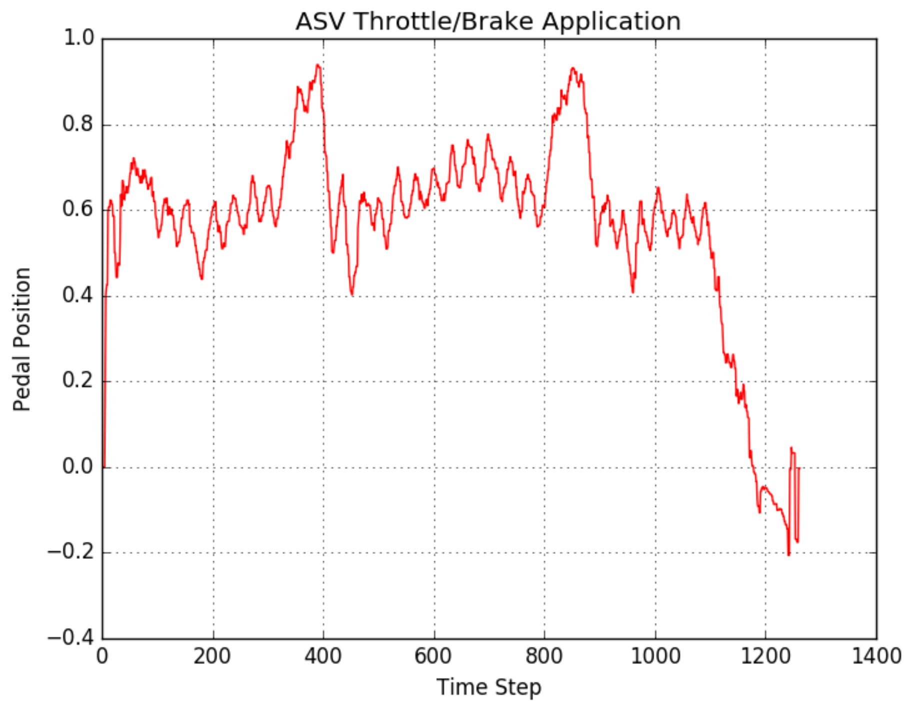


Figure 4.15: Throttle/Brake application of the Tesla on highway exits.

Using the learned models, the controller was able to stabilize the system at different road gradients. However, the tested scenario took place in the same condition as the condition when the data was collected. Even though the MPC performed well, no conclusion can be made regarding the performance of the MPC under different conditions. More data needs to be collected to put more trust into the system.

4.3.2.2 Highway Driving

This scenario was designed to test the ability of the controller to handle high tangential velocities at the limits of lateral acceleration. The test scenario is 1 km in length with speeds exceeding 25 $\frac{m}{s}$ (56 mph) on the highway. Figures 4.16 and 4.17 show the Tesla's MPC tracking performance and the cross-track error during testing.

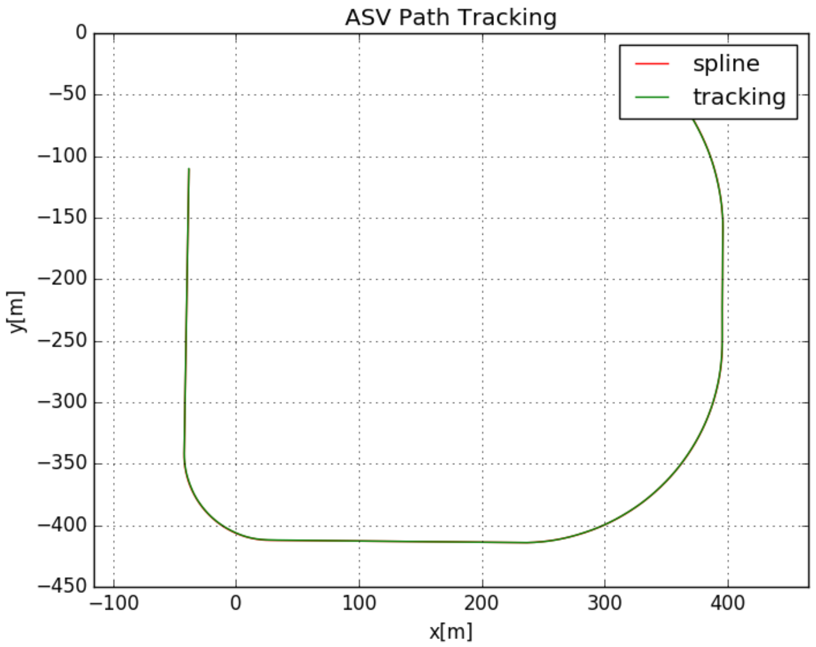


Figure 4.16: Tracking performance of Tesla with the Nonlinear MPC for highway driving.

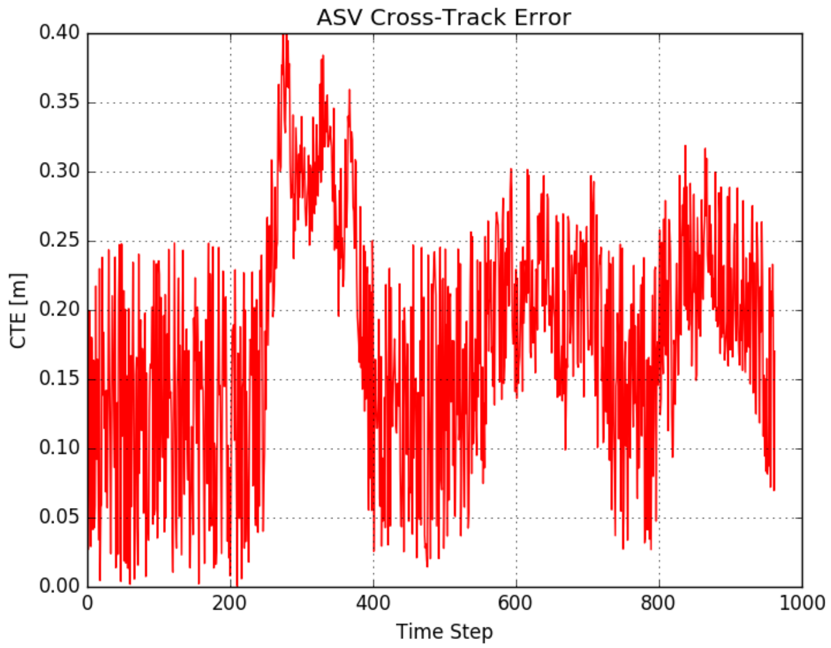


Figure 4.17: Cross-track error of the Tesla on the highway.

The MPC had a cross-track error average of 0.178 m and a standard deviation of 0.084 m. The MPC was able to handle the weight transfer encountered at high speeds. However, Figure 4.18 shows that the longitudinal controller struggled to stabilize the speed of the vehicle at 20 $\frac{m}{s}$ and at 2.5 $\frac{m}{s^2}$ lateral acceleration. The overdamped behavior of the correction term in the longitudinal controller can be observed when there is an abrupt change in the reference velocity.

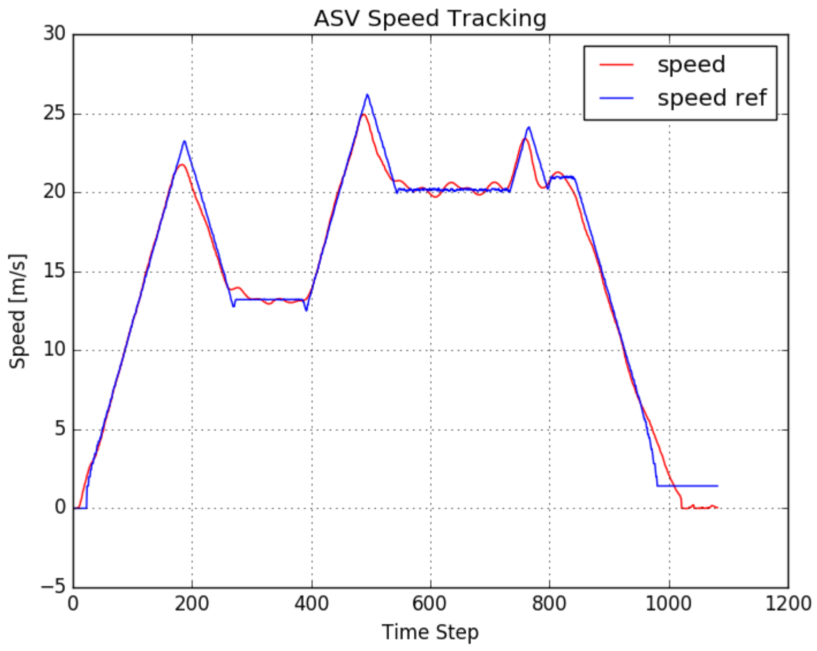


Figure 4.18: Velocity tracking performance of the longitudinal controller on the highway.

Figure 4.19 displays the heading angle tracking performance of the MPC. The MPC successfully stabilized the Tesla without any swaying motion. The mean heading angle error average was 0.0086 rads and a standard deviation of 0.00867 rads. Figure 4.20 illustrates the pedal application on the highway.

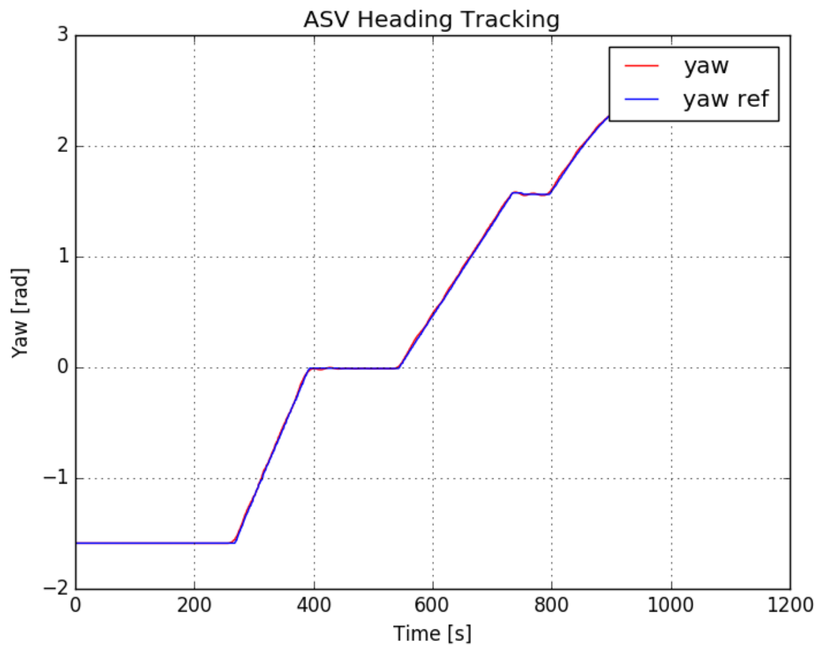


Figure 4.19: Yaw tracking performance of the MPC on a highway.

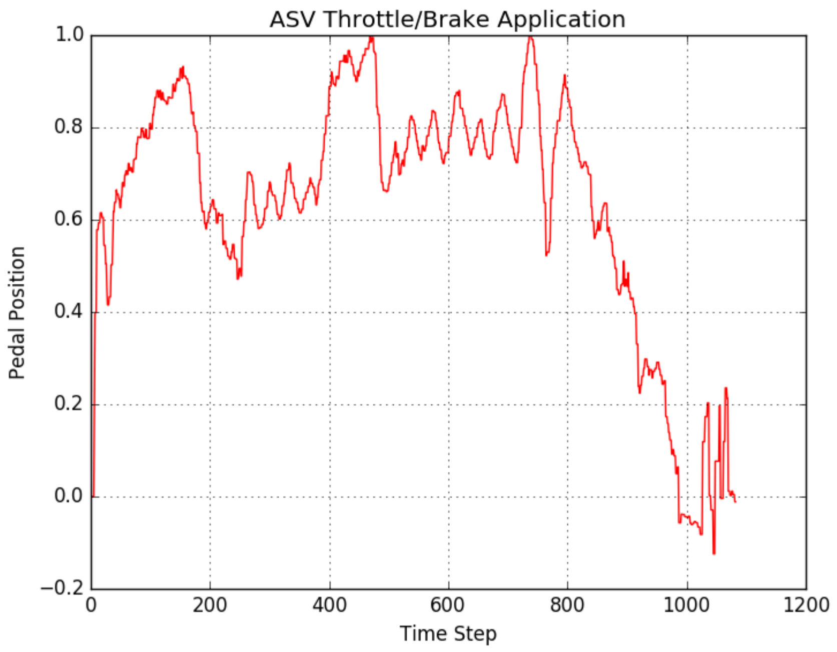


Figure 4.20: Throttle/Brake application of the Tesla on a highway.

4.3.2.3 City Driving

The city driving scenario was designed to test the controller during high curvature, high lateral acceleration maneuvers. The scenario is 1 km long with 90° turns and a roundabout. Figures 4.21 and 4.22 shows the tracking performance of Tesla's lateral controller and the cross-track error over time, respectively. This test reveals the downsides of the MPC. For 90° turns, the MPC fails to stabilize the system. However, external factors could have effected the MPCs performance. The way the paths are generated does not account for the dynamics of the vehicle. Therefore, the paths may not be suitable for a vehicle to track. Another factor that can effect the tracking performance is that the steering rate of the vehicle might not be fast enough for such maneuvers.

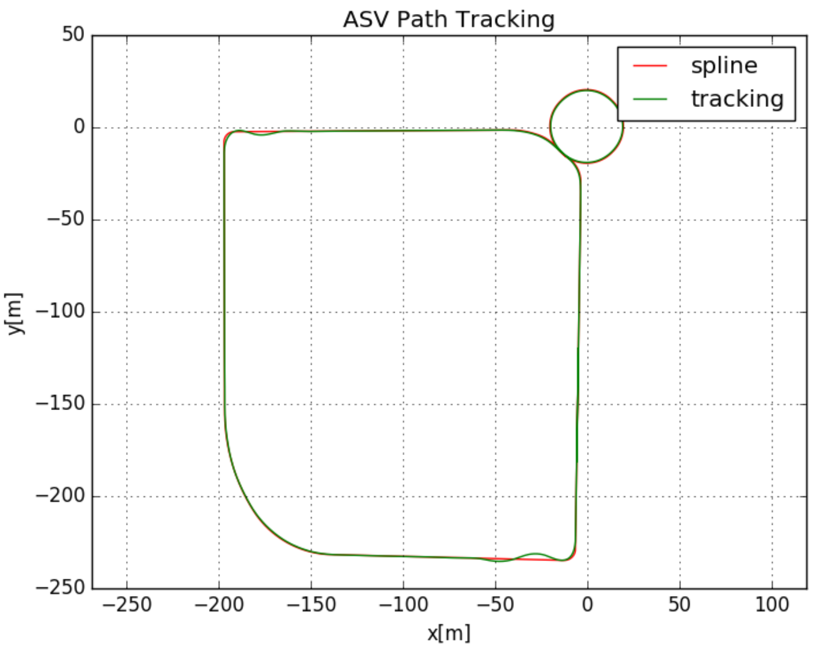


Figure 4.21: Tracking performance of Tesla with the Nonlinear MPC for city driving.

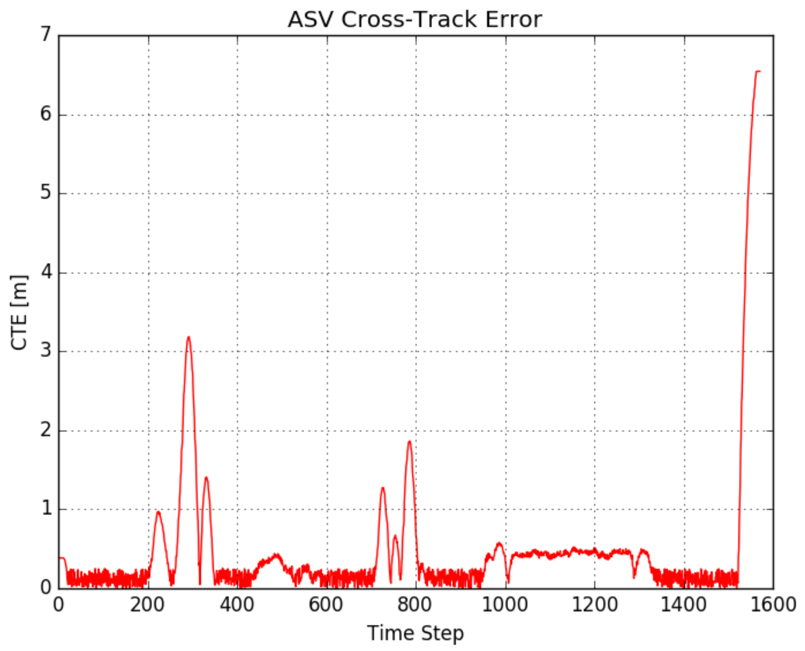


Figure 4.22: Cross-track error of the MPC for city driving.

The average cross-track error for city driving is 0.489 m with a standard deviation of 0.925 m. The controller cannot be safely deployed for city driving as the cross-track error exceeded 3 m on the first turn. Assume the average car width to be 1.9 , and the lane width is 3.7 m, this allows the driver to have 0.9 m on each side of the vehicle. The controller failed to keep the vehicle within the lanes. In the roundabout, the controller lowered the speed of the vehicle to lower the cross-track error, Figure 4.23. This shows that CTE_v and θ_v are behaving as expected in the objective function. Figure 4.24 clearly shows that the controller faces difficulties when heading angle rate is high, which occur around time-step 300 and 800.

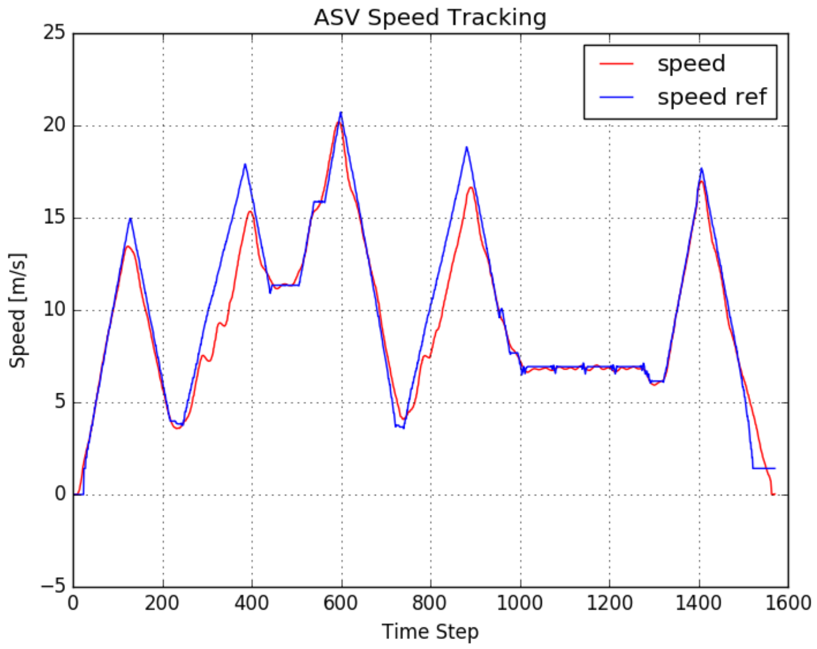


Figure 4.23: Velocity tracking performance of the longitudinal controller for city driving.

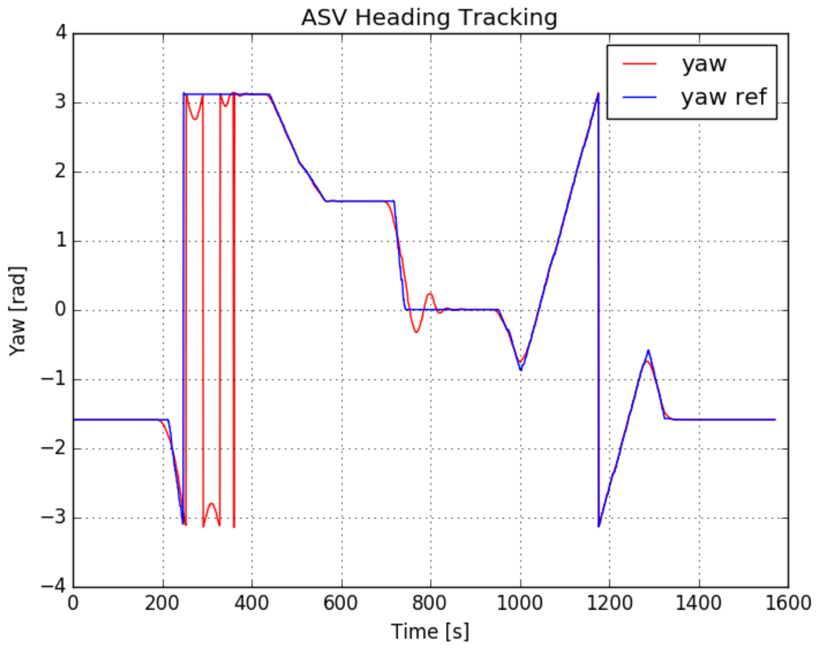


Figure 4.24: Yaw tracking performance of the MPC on a highway.

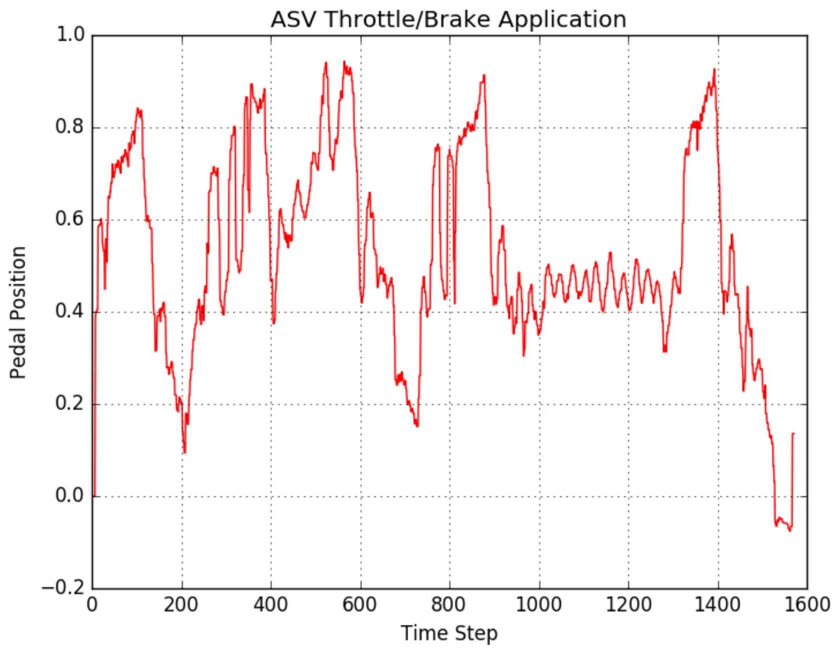


Figure 4.25: Throttle/Brake application of the Tesla in a city.

5. CONCLUSION AND FUTURE WORK

5.1 Conclusions

In this thesis, a general purpose data-driven approach to system identification for the Warthog and ASVs was proposed to improve on the physics-based modeling. In most cases, the data-driven approach outperformed the physics based model. A Nonlinear MPC was also proposed to control the aforementioned systems for path tracking.

First, data was collected from the Warthog by driving it around the RELLIS Campus using a remote control. Approximately 17,000 odometry sample points were used to train and evaluate the machine learning models. The results show that the L_2 regularization did not improve the predictive models, and it would sometimes hurt the accuracy. Even though the results show that the LASSO regression outperformed the physics model, the only thing that can be said is that it only outperforms the physics-based model under similar conditions to when the data was collected.

Same thing was done for the ASV, however, the data collected was in a simulator called CARLA. The system identification slightly improved the positional estimate of the vehicle, however, it improved the heading angle model by 11%. Similar findings were found when compared to the Warthog; the L_2 regularization did not improve the prediction model. The models that were obtained through the data-driven system identification, suggested that there might be a small delay in the system that has not been accounted for by the physics-based model.

The controllers were then evaluated using open source simulators; Gazebo and CARLA. Depending on the system, the objective function and the constraints were designed accordingly. The main goal for the Warthog is to track a path as accurately as possible. For the ASV, path tracking is also important but with the added passenger comfort constraints on the lateral and longitudinal acceleration. Warthog's MPC tracked the path with a cross-track error of 8 cm. During turns, the MPC struggled with Warthog losing traction but it was stable for the entire path.

The ASV controller faced more difficult challenges; higher lateral and longitudinal accelerations, lower control sampling rate, and higher inertia. Three testing scenarios were created to test specific aspects of the controller. The first test was highway exits and entrances. The goal is to test the controller in different road gradients. The controller succeeded to stabilize the vehicle with a cross-track error of 0.28 m, which well within the safety margin. This test also showcased the proposed longitudinal controller in action. The correction term, $\sum \text{PID}(v_{error})$, was able to track the velocity reference during the uphill and downhill segments of the test. However, it was found that the controller was overdamped regardless of the tuning. The second test was conducted on a highway. The goal was to test the MPC during high speeds and high load transfers. It followed the lane with an accuracy of 0.084 m accuracy with speeds reaching 56 mph; the speed was limited by the lateral acceleration. During testing, the velocity tracking error was the most important factor to be tuned in the objective function. The last test was the city driving. The goal is to test the controller during high curvature turns. This test shows that the controller can not be tuned for all test case scenarios. The controller failed to stabilize the vehicle for 90° and failed to keep the vehicle within the safety margin of the lane. Many factors were identified that could have effected the results. The path generated by the simulator did not account for the vehicle dynamics, thus making it difficult track. The second reason is that the control input rate, mainly the steering rate, is not fast enough to track the trajectory at 2.5 $\frac{m}{s^2}$ lateral acceleration.

While researching, the data-driven approach to system identification and controls showed promising results. It gave great insights, such as delay in the system, for the dynamic system. The proposed approach is also effective and applicable for any system.

5.2 Future Work

Feature Engineering: The current method of collecting data does not support any conclusions made about generality of the models. There is no quantitative method that can describe how diverse or how much of the dynamics of the system was explained by the collected data. A metric should be developed to answer both of those questions. The proposed method uses domain knowledge which yielded good results, but what kind of models would have been generated if no such assumptions were made?

System Identification: The work done in this thesis solely focuses on time-domain system identification. A frequency-domain approach may yield better results as the model can account for noise and disturbances. Kinematic models were used to extract domain-knowledge based feature sets. Dynamics models should be considered as they can provide a feature set that accounts for system inertia.

Controller Design: Tuning the controllers proved to be the most difficult aspect of the controller design. It heavily influenced by the parameters of the system, the trajectory for the path and the velocities, external disturbances, and uncertainty of the system. Developing a tuning mechanism that accounts for all these factors can cut down developing time as well as guarantee, to a certain extent, the performance of the controller. Dynamic systems in the real-world are uncertain, developing a stochastic MPC is expected improve the results.

REFERENCES

[1] J. C. Bebel, N. Howard, and T. Patel, “An autonomous system used in the darpa grand challenge,” in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, pp. 487–490, 2004.

[2] M. I. Palmqvist, “Model predictive control for autonomous driving of a truck,” 2016.

[3] 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)*, pp. 1094–1099, 2015.

[4] Z. Ercan, M. Gokasan, and F. Borrelli, “An adaptive and predictive controller design for lateral control of an autonomous vehicle,” in *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 13–18, 2017.

[5] J. Hu, S. Rakheja, and Y. Zhang, “Real-time estimation of tire–road friction coefficient based on lateral vehicle dynamics,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 234, no. 10-11, pp. 2444–2457, 2020.

[6] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, “Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing,” in *2007 American Control Conference*, pp. 2296–2301, 2007.

[7] A. AbdElmoniem, A. Osama, M. Abdelaziz, and S. A. Maged, “A path-tracking algorithm using predictive stanley lateral controller,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 6, p. 1729881420974852, 2020.

[8] N. H. Amer, K. Hudha, H. Zamzuri, V. R. Aparow, A. F. Z. Abidin, Z. A. Kadir, and M. Murrad, “Adaptive modified stanley controller with fuzzy supervisory system for trajectory tracking of an autonomous armoured vehicle,” *Robotics and Autonomous Systems*, vol. 105, pp. 94–111, 2018.

[9] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” Tech. Rep. CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA, January 1992.

[10] G. Pannocchia, J. B. Rawlings, and S. J. Wright, “Inherently robust suboptimal nonlinear mpc: Theory and application,” in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 3398–3403, 2011.

[11] U. Rosolia and F. Borrelli, “Learning model predictive control for iterative tasks. a data-driven control framework,” *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2018.

[12] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.

[13] G. Wang, Q. S. Jia, J. Qiao, J. Bi, and M. Zhou, “Deep learning-based model predictive control for continuous stirred-tank reactor system,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2020.

[14] G. L. Woods, “Model predictive control for autonomous driving of a truck,” 05 2020.

[15] J. Brownlee, “Discover feature engineering, how to engineer features and how to get good at it,” Aug 2020.

[16] R. Karim, “Intuitions on l1 and l2 regularisation,” Oct 2020.

[17] M. I. Ribeiro, “Kinematics models of kinematics models of mobile robots.”

[18] G. Deol, “An introduction to ridge, lasso, and elastic net regression,” Feb 2019.

[19] A. Chakure, “Random forest and its implementation,” Nov 2020.

[20] V. Morde, “Xgboost algorithm: Long may she reign!,” Apr 2019.

[21] L. Chlon, “tl;dr: Gaussian process bayesian optimization,” Jan 2020.

[22] V. M. Becerra, “Optimal control,” 2008.

[23] V. Yadav, “Direct collocation for optimal control,” 2016.

[24] I. Bae, J. Moon, and J. Seo, “Toward a comfortable driving experience for a self-driving shuttle bus,” *Electronics*, vol. 8, no. 9, p. 943, 2019.

[25] Y. Peng, J. Chen, J. Yu, Y. Ma, and H. Zheng, “Nonlinear observer for vehicle velocity and tire-road friction coefficient estimation,” in *2017 American Control Conference (ACC)*, pp. 2606–2611, 2017.

[26] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, 2004.

[27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] 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, p. 1–36, 2018.

[30] F. E. Curtis, J. Huber, O. Schenk, and A. Wächter, “A note on the implementation of an interior-point algorithm for nonlinear optimization with inexact step computations,” *Mathematical Programming*, vol. 136, no. 1, p. 209–227, 2012.