12-1-2018

Implementation of a brassboard prototype of a collision avoidance system for use in ground vehicles

Tyler James Hannis

Implementation of a brassboard prototype of a collision avoidance system for use in ground vehicles.

By

Tyler James Hannis

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2018

Implementation of a brassboard prototype of a collision avoidance system for use in ground vehicles.

By

Tyler James Hannis

Approved:

_____

John E. Ball
(Major Professor)

_____

Jian Shi
(Director of Thesis)

_____

John A. Hamilton, Jr.
(Committee Member)

_____

James E. Fowler
(Graduate Coordinator)

_____

Jason M. Keith
Dean
Bagley College of Engineering

Name: Tyler James Hannis

Date of Degree: December 14, 2018

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Director of Thesis: Jian Shi

Title of Study: Implementation of a brassboard prototype of a collision avoidance system for use in ground vehicles.

Pages in Study 65

Candidate for Degree of Master of Science

Accidental collisions involving wheeled industrial ground vehicles can be costly to repair, cause serious (even fatal) human injury, and lead to setbacks with tight operation schedules. Reduction of vehicle collisions carries numerous safety and financial incentives. In this work, an integrated collision avoidance package is developed to reduce the number of vehicle collisions. Utilizing feedback from on-board sensing devices, a model predictive control (MPC) algorithm predicts control options and paths, then disallows drivers to accelerate and/or induces braking of the vehicle if a collision is imminent. A prototype system is developed, implemented, and tested on an industrial vehicle to mitigate collisions with people and high-value equipment. Testing results show that control can be executed in real time by the proposed system, and that the proposed method is effective in preventing an industrial vehicle from hitting detected obstacles and entering restricted areas.

ACKNOWLEDGEMENTS

I would like to thank my committee members Dr. John Ball, Dr. Jian Shi, and Dr. Drew Hamilton who have helped support my studies. I would like Jim Gafford for his professional mentorship, and my coworkers who have helped support my research. I would also like to thank my friends and family members who have encouraged me throughout years. Finally, I would like to thank my wife Melissa for her continued love and support for my career pursuits.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation and Overview

One applications of cargo tractors are to utilize multiple sequential trailers behind one tractor to reduce the number of vehicles needed to haul a large quantity of supplies or workloads to new locations. Navigation of such a system is complicated and can be prone to human error, especially in dynamic environments. This type of vehicular system is often operating nearby expensive equipment and areas of restricted ingress. Accidental collisions involving wheeled industrial ground vehicles can be costly to repair, cause serious (even fatal) human injury, and lead to setbacks with tight operation schedules. In this work, a prototype system to help prevent a tractor-trailer system from making accidental collisions is developed, integrated into a vehicle, and tested for its efficacy.

1.2 Collision Avoidance System

System components such as a vehicle model, sensor package, obstacle detection method, predictive control method, and user control overrides will be integrated together to compose a system capable of preventing a vehicle collision. A real-time, control based, autonomous Collision Avoidance System (CAS) will seek to integrate functions as summarized below.

1.2.1 Environment Understanding

To introduce a protective system such as CAS into any dynamic system, some means of relating the system to its environment must first be established. For ground vehicles, this type of environmental sensing can be realized by observing anything that may impede the system's ability to traverse its course. Additionally, the system should have regard for anything that might not impede the vehicle but still cause damage in the event of a collision. In either case the protective system should handle detections safely and reliably.

Collision avoidance systems need a precise understanding of the world surrounding the vehicle(s) in question. The authors of [1] performed research into accounting for a dynamic environment by utilizing a Light Detection and Ranging (LiDAR) sensor to provide information to an autonomous vehicle controller about the system's changing world model. This form of obstacle detection and environment identification can be expanded to various other sensors such as cameras, ultrasonic sensors, and (but not limited to) positioning methods such as Global Positioning System (GPS) [2].

Once the environment is mapped, the vehicle can be projected into the environment. A control algorithm can then consider the current state of the vehicle, establish valid paths, and then execute an optimal control solution. This collision avoidance control solution can manifest itself as either slowing down the vehicle speed or bringing the vehicle to a complete stop. This is due to the lack of capability of controlling steering of the vehicle. The vehicle speed and nearness of discovered objects will have the largest impacts on the control solution generated.

1.2.2 Data Feedback from Sensors

If an autonomy-equipped ground vehicle is to perform successfully in a desired environment, it must be able to understand the world around the vehicle. One way to facilitate this is for the ground vehicle to be equipped with a LiDAR package as well as a Red-Green-Blue (RGB) camera that helps with localization of recognized objects in the 3-dimensional space around the vehicle [3]. Quick and reliable readings from the 3-dimensional space around the vehicle are essential to being able to develop a control algorithm capable of keeping a vehicle away from areas and objects that are desired to be avoided. Additionally, vehicle dynamics readings such as the vehicle velocity, acceleration, and heading will be used to model future potential paths the vehicle could take. In combination with a representative vehicle model, this data will allow for optimizing potential vehicle trajectories.

It should be noted that there are many ways to go about reading a vehicle's state; the method described above will be used for this work.

1.2.3 System Virtual Modeling

A physics based representative model of a ground vehicle and subsequent trailers was previously created and used in vehicle trajectory simulation [4]. This model can be provided with vehicle information like velocity and acceleration, which it then uses to propagate a virtual model of where the vehicle's current and future locations. The CAS can utilize this model by optimizing model inputs and determining if they allow for valid vehicle movement. If the model inputs provided allow for a driver to hit a detected object, the controller will react and try to prevent this detected collision.

This developed model includes capabilities to track up to four trailers pulled by the vehicle, as four trailer systems are typically used in the application being developed for. These tractors are counted as a part of the vehicle system and used by the CAS to track where the vehicle is. Each trailer, along with the vehicle, is represented as an interconnected polygonal model shown in Figure 1.1 below [4]. This vehicle system is comparable to a train in the sense that each body is connected in the front and/or back to another body. The properties in how the trailers follow the path of the vehicle have been well modeled and prove accurate assuming the system is driving forward. The CAS system will not be designed to handle reversed movement as the modeling and controller are being designed for forward movement only. For this development, the vehicle and trailers polygonal models will be used when predictively keeping the vehicular system away from undesired areas.

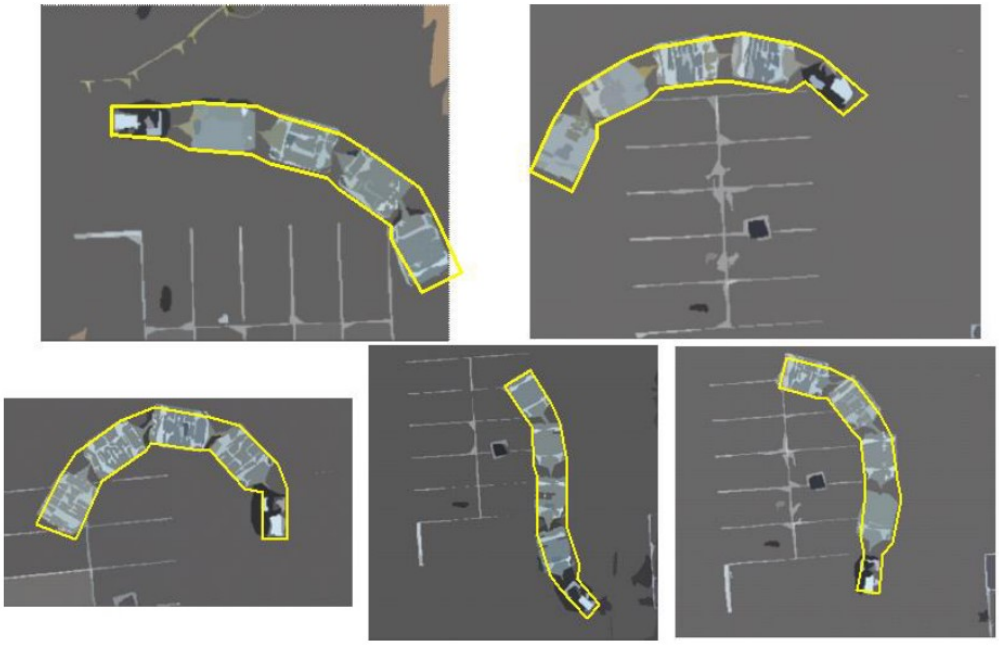

Figure 1.1 Figure demonstrating the polygonal representation of the vehicle trailer system as demonstrated in [4].

1.2.4 Integration of the System into a Vehicle

To add the needed degree of intelligence to the system for it to operate in real time conditions, the vehicle will be modified to include the sensors mentioned in section 1.2.2 above and also house other necessary computational systems. The first of these systems will process the LiDAR and camera data to report back the detected objects around the vehicle [3]. The other system will use this detection data as well as the vehicle dynamics readings as inputs to a Model Predictive Control (MPC) scheme to evaluate the vehicles potential trajectories and ensure the vehicle is not allowed to make a collision with a detected area or object. Additionally, a microcontroller will be installed to enact braking and throttle control when appropriate. This refit is depicted in Figure 1.2 below.

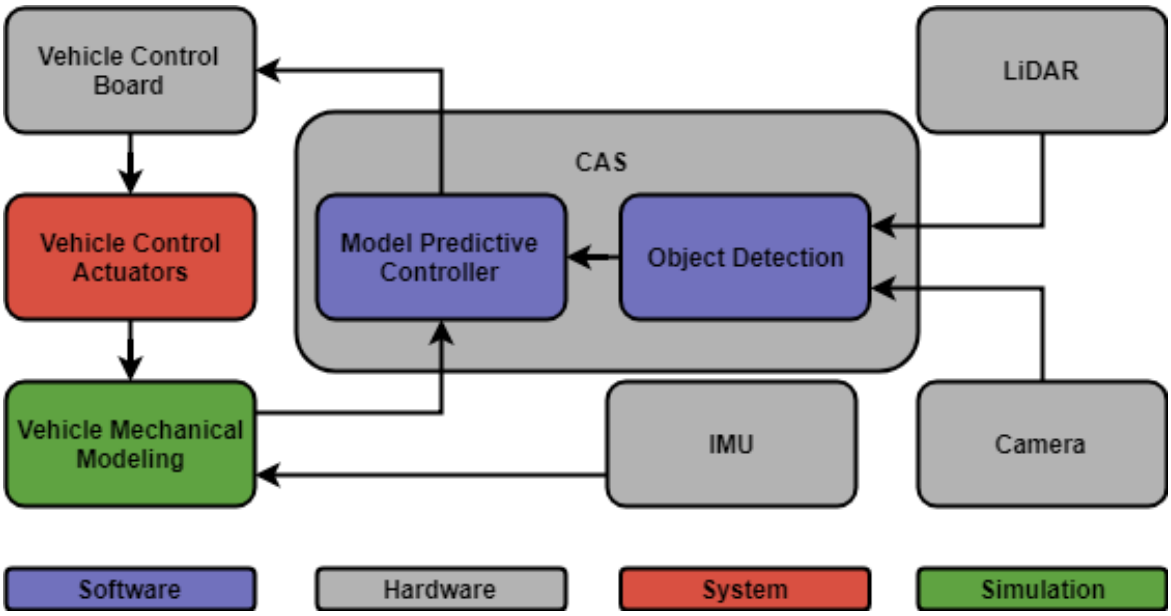

Figure 1.2 Depiction of the integrated control package for the vehicle. Image is best viewed in color.

Environmental data is captured from the LiDAR and camera system. This data is provided to an object detection package. Vehicle data including speed, acceleration, and heading are given to the mechanical modeling package described in section 1.2.3 above. After the MPC receives both updates, it will provide control instructions to the vehicle control board detailed in section 1.2.5 below.

1.2.5 Actuation Control

To accomplish the needs of the control system, a microcontroller will enact the control policies of the control algorithm in real time. The controller will interface with the vehicle by pairing with driver input controls such as the gas pedal and brake pedal. Using various control techniques, the controller will override and/or adjust driver input to ensure the driver is not able to take a trajectory rejected by the CAS. The actuation controller has a direct communication line to the collision avoidance controller to

minimize the latency of exchanging system dynamic data and receiving new control inputs from the CAS.

1.3 Contributions

The novelty of this collision avoidance package retrofitted in the vehicle platform were focused on core MPC algorithm development and refinement, and mechanical control development. I also handled the communications between these two platforms and helped with the installation and integration of the new hardware.

1.3.1 MPC Algorithm Development

An MPC algorithm was first tested in simulation and then implemented after its validity was proven. Performance between different algorithms were evaluated and iteratively adjusted until acceptable functionality was achieved. These iterations included updating the physical rules the vehicle system is bound by and improving the computational performance of the system to hit real-time goals. These iterations include

The MPC was designed to be able to simulate the vehicle and trailer movement in a 2-dimensional plane. It is passed live updates of the current vehicle state during the beginning of the MPC calculations then propagates the estimated states. The MPC is designed such that the most freedom possible is given to the vehicle driver that also does not allow for a collision or crossing a restricted boundary. This approach is taken to ensure that the CAS package is not a productivity hinderance but also capable of accomplishing its designed goals.

The algorithm controls off vehicle speed such that a speed is never allowed to be traveled by the vehicle that could result in the vehicle hitting a detected object or entering

a restricted perimeter. Since heading (or steering angle) was not a controlled variable, the MPC works on the assumption that the driver will intentionally take the most direct route that will result in a collision. The foundation of this reasoning is if the controller can prevent the most likely course of action that can cause an accident, other less likely actions are inherently included in the control solution.

1.3.2 Mechanical Control Development

The CAS installed vehicle must have some means of allowing the CAS system to take control. This was accomplished by installing a dedicated control board to interact with the vehicle control (like the gas pedal and brake pedal). This actuation controller implementation was selected such that it can be guaranteed that control actions would be performed in real-time without any control delay caused by CAS load. This secondary controller can also account for lack of communication or failure of the CAS system, allowing the vehicle to be operable even if the CAS is inactive or malfunctioning.

The actuation controller intercepts the signal produced by the gas pedal and reproduces this input to the vehicle's Electronic Control Unit (ECU). This is necessary so that control inputs from the MPC can be enacted to directly override drive input. When a driver provides control inputs, the actuation controller allows these controls to be passed through as long as the input is at or below the threshold needed to ensure the vehicle remains below the target speed set by the MPC algorithm. If the driver attempts to send a control input greater than the MPC threshold, the actuation will reduce the input only allowing control passthrough that ensures the vehicle stays at or under its target speed.

Additionally, the actuation controller is capable of enacting vehicle brakes when a speed reduction or emergency stop is deemed necessary by the CAS system. This is

accomplished by an installed brake actuator driven with the actuation controller. When the actuation controller receives the signal to brake, a deceleration target is accompanied with that signal indicating how aggressively the vehicle must slow down. When the controller is braking, input from the gas pedal is ignored until the system achieves its new speed target. As an additional benefit of having an installed external brake actuator, a brake assistance method was created to help the driver brake with significantly less force enacted on the brake pedal. Since a loaded vehicle-trailer system can have a large momentum when slowing down, this addition made driver induced braking easier as an extra feature.

1.3.3 Hardware Installation and Testing

The CAS tractor had to be fitted with several new and modified systems, and each change had to be able to sustain in the rugged operating environment the tractor is operated in. I helped in the development and installation of the new and modified equipment. Each wiring harness was secured, and the ends made weather tight. The computing platform was installed in a vented, weather sound enclosure. The mechanical controller was fitted in a sealed enclosure and discretely mounted on the vehicle. All of these and more considerations were accounted for to ensure that the system could reliably hold up to the environment.

The CAS system required several hours of tuning and validation. This involved evaluating the performance of the mechanical controls to ensure that the system reacts as expected when directed by the CAS controller. Also, evaluation of the controlling MPC algorithm was required, and several validation tests had to be checked each modification

of the algorithm to ensure validity. I also assisted in installing, testing, and validating the sensor package for the vehicle, though I did not contribute to the software development.

CHAPTER II

BACKGROUND AND DESIGN FOCUSES

2.1 Collision Avoidance

With the advancements made towards autonomous vehicles, a reliable means of protecting and safeguarding these vehicles have been sought after and researched. The topic of collision avoidance has manifested itself as one of the most effective but challenging strategies for protecting enhanced or autonomous vehicles. Technological advances, such as faster data processing better sensing capabilities, have allowed sophisticated obstacle detection and vehicle tracking to be integrated in union to create realistic means of implementing various predictive algorithms into vehicular systems. Analysis of using a lesser and greater understanding of vehicle environment has been done by evaluation of algorithm performance with varying grades of vehicle models [5].

The area of focus for collision avoidance is vast, extending from industrial equipment integration [6] to personal transportation [7] to even robotic applications [8]. Whether being integrated into fully autonomous vehicles or even driver assistance systems, reliable collision avoidance techniques can be cost saving and improve the operational safety of the system and surrounding environment. Additionally, the installation of a CAS is worth investing in due to their improved costs and time savings from damages caused by accidents.

Collision avoidance can be causally related to path planning approaches. While collision avoidance is trying to prevent a system from entering into an area (usually to prevent hitting an object), the other is directing a system into an area. It is common to see these principles being combined in research works [9]–[11]. Maintaining a designated route with path planning requires the same understanding of the vehicle environment, such as seen in collision avoidance algorithms [5]. Typically, a minimally acceptable distance to stray away from detecting obstacles is defined [12] and is dynamic function of the vehicle's speed. This distance is usually optimized to account for a vehicle's ability to brake, and large enough not to overly restrict the controlling algorithm to ensure that the controller can find an acceptable solution in real time.

2.2 Model Predictive Control

MPC is a control scheme that estimates the future system state. This discrete method has various applications throughout physical and digital systems [13]–[17]. In physical systems, MPC algorithms offer advantages over reactive controllers such as the common Proportional–Integral–Derivative (PID) controller [18], [19]. Though much more computationally demanding, the MPC algorithm can support finite control solutions and account for delays that are not supported by reactive controllers [20]. The core structure of an MPC algorithm includes a model representing the system being controlled, a representation of past control inputs and a cost function that allows for intelligent control optimizations to achieve the desired system reaction.

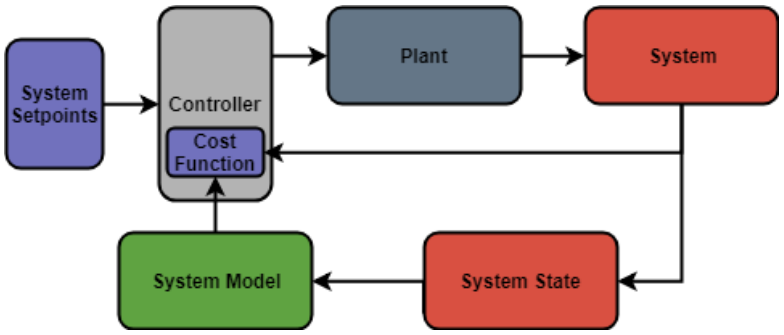

Figure 2.1 Structure of a generic model predictive control implementation.

Nonlinear MPC algorithms have emerged for use in vehicle path planning [21], [22], relying heavily on model representations of the environment and vehicle. Regarding path planning algorithms, search algorithms (like Limited Lookahead Control) can be used to optimize the control solution [23], [24]. The complexity of this form of optimization varies on the number of future steps the controller is trying to predict. Additionally, the complexity of the controller increases with the number of parameters the algorithm is trying to optimize. In the application of obstacle avoidance, the controller will often have to account for more than just vehicle speed to be able to efficiently prevent collisions. Speed control can be optimized with an autonomous vehicle to be able to prevent collisions and plan paths [25]. Alternatively, the controller should be able to at least account for the variables in the system that it cannot control and operate on the rest of the system to prevent an undesired outcome. Nonlinear MPC algorithms have emerged for use in vehicle path planning[21], [22], relying heavily on model representations of the environment and vehicle. Regarding path planning algorithms, search algorithms can be used to optimize the control solution. The complexity of this form of optimization varies on the number of future steps the controller is trying to

predict. Additionally, the complexity of the controller increases with the number of parameters the algorithm is trying to optimize. In the application of obstacle avoidance, the controller will often have to account for more than just vehicle speed to be able to efficiently prevent collisions. Control of steering and speed can be optimized with an autonomous vehicle to be able to prevent collisions and path plan[25]. Alternatively, the controller should be able to at least account for the variables in the system that it cannot control and operate on the rest of the system to prevent an undesired outcome.

MPC can be very computationally intensive [26], [27], so to be able to achieve real time performance in the controlled system, the computation platform should be selected around the specific control application. Factors affecting the controller performance include the number of parameters being acted on, the number of control options the algorithm can try to optimize, and the prediction horizon of the controller [28]. This detail of optimization greatly expands the capability of the controller but can introduce latency into the system that should be accounted for. The algorithm must be able to converge on a candidate solution before the system needs to execute said solution. If this is unable to be achieved, the controller may not be able to prevent an operating vehicle from making a collision, for example.

Investment should be made when creating an MPC controller to optimize the amount of predictive insight provided while also maintaining stable and reliable control operations. Trying to predict extremely far ahead can become computationally taxing quickly [26]. When a system has a wide range of control inputs, extrapolating these options into future timesteps could be slow [29]. While having a well evaluated prediction horizon is desirable, an MPC controller will most likely have some defined

time constraints that it must be able to complete its predictions and optimizations within. If a wide choice of control inputs is desirable, then a narrower prediction horizon should be targeted to ensure responsiveness of the controller.

2.3 System Requirements

The scope of this study was to evaluate the effectiveness of control approaches and the installation of a CAS on an industrial tractor/multi-trailer vehicular system. As part of a previous research project, the stock tractor in question was retrofitted with a sensor package providing extended system metrics such as speed, steering angle and other metrics not related to this study. The controller uses this data so that it can have a real-time understanding of its movement dynamics in the surrounding environment. It would be impossible to predict future outcomes of vehicle travel without live updates of the vehicle and environment states.

A combination of sensors including LiDAR, an RGB mono-camera, and a nine-Degrees of Freedom (DOF) Inertia Measurement Unit (IMU) were also installed on the tractor system. The information obtained from these additions are necessary for research into the collision avoidance methods. An image detailing the mounting location and integration of the LiDAR sensor can be seen in Figure 2.2 below.

Figure 2.2 Image of mounted LiDAR sensor on raised bracket, providing 270° coverage around the front of the tractor. Sensor circled in red. Figure best viewed in color.

The LiDAR is mounted off-center to leave the driver view as unobstructed as possible.

The control algorithm needs to be tuned and calibrated to produce collision free control actions. The safest control action that guarantees no collisions results in always keeping the vehicle at a stop. This would be frustrating to the user(s) of a protected vehicle, so the controller should allow the most amount of system freedom while also preventing undesired paths for being traversed. A balanced compromise of freedom to move and yielding to hazards are ideal components in a vehicular protective system.

It is expected for the CAS system to be able to keep the vehicle and following tractors from hitting detected objects or entering designated restricted areas. This is accomplished by only allowing driver control actions to be enacted if they do not violate this rule set. When the CAS determines a violation the user control inputs are rejected and/or altered such that the constraints are met. The system must be able to react quickly

enough to ensure that the physical system can also respond. Since it is expected for this system to have multiple trailers conjoined and possibly loaded with cargo, the momentum of the system makes for a large force to counteract.

2.4 Design Strategy

This work focuses on being able to provide minimally restrictive interference to a tractor driver by controlling the speed when the vehicle is in imminent danger making a collision. The controller will be able to consider the information provided by the various sensors mentioned. Beacon positions are reported as a function of distance from the tractor and angle relative to the tractor, and the controller understands its nature of movement in the environment with the inertial related sensors installed. Using this information, the controller uses an iterative predictive approach to find optimal control candidates that allow for the most amount of unrestricted travel to the tractor driver, while also preventing the driver from being able to enter a designated restricted area.

To be able to properly design the controlling algorithm, it is important to understand the nature of control the algorithm will have, and its limitations. The designed controller must be able to dynamically adjust the tractor's maximally allowed travel speed. An actuation controller on the tractor will limit the vehicle speed if the driver is or attempting to travel faster than the controller's maximum speed. The actuation controller will always allow the tractor driver to drive below the collision avoidance controller's maximum allowed speed. The algorithm is a driver assistance package, meaning that it is not an autonomous solution capable of self-driving. The driver on the tractor always has full control over the tractor's trajectory, with the controller not being able to adjust the

steering angle. Therefore, the control algorithm must consider potential changes in trajectory to ensure the driver does not get the tractor into an unpredicted state.

2.4.1 Object Detection

A portable area marker was created such that it could easily be detected by the sensor package on the vehicle. This design includes having a long pole equipped with retroreflective tape creating a unique shape that the LiDAR can easily see. Another part of the design included the base being constructed out of an orange pylon. This adds structural support and another easily distinguishable trait for object detection. An example of this marker can be seen in Figure 2.3 below. These markers are designed such that several beams from the LiDAR can reflect off the tape creating a feature rich detection. The LiDAR used has eight scanning beams spaced approximately 3° apart from each other. The tape is vertically aligned such that multiple beams intersect the tape returning an outstanding shape to the object detection algorithms.

These markers, hereby referred to as beacons, are used to fence off areas where the tractor and subsequent trailers are not allowed to travel in. This is accomplished by the MPC algorithm keeping the system a minimum distance away from beacon detections. If several beacons are placed together they can create a pseudo-fence around an obstacle or area the tractor should not enter. It should also be noted that the obstacle detections algorithms are not limited to strictly seeing these beacons. Larger objects such as walls or even nearby pedestrians in the front area of the vehicle are recognized for the system to prevent striking. This detection, herein referred to as front guard, works at a width of three meters in front of the vehicle extending up to 20 meters from the front of

the vehicle. The ability to detect these frontward objects is variant on the reflectivity of the object and material in front of the system.

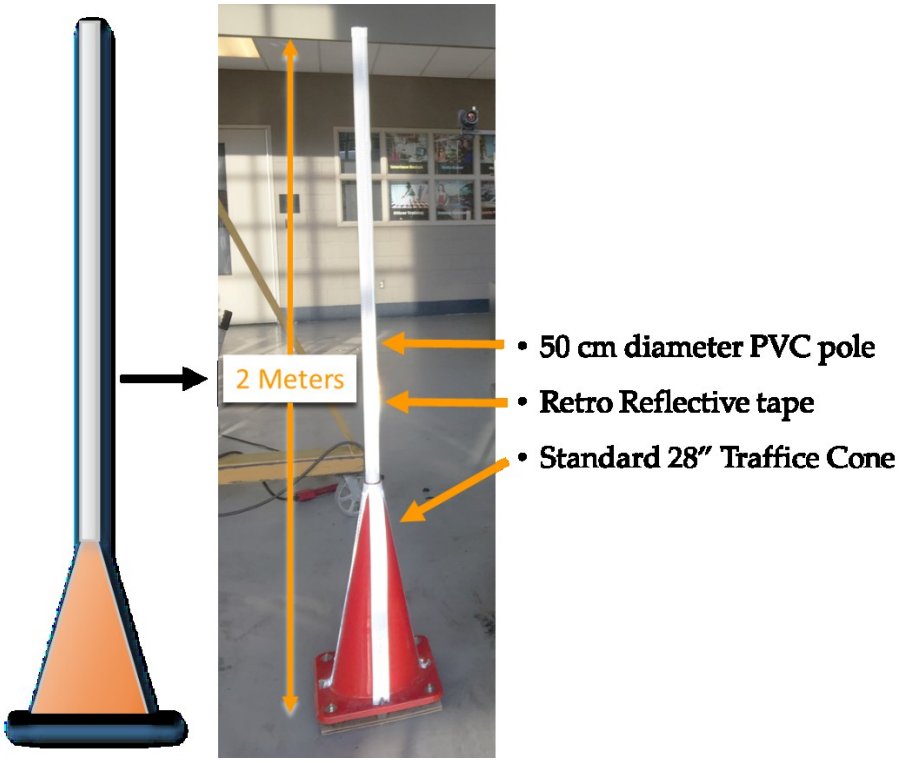

Figure 2.3 Image of a retroreflective beacon.

Additionally, an onboard camera sampled image data from the area in front of the vehicle. This camera was integrated into the front bumper of the vehicle as pictured in Figure 2.3 below. The camera captures were used in correlation to the LiDAR readings to help verify object detections and eliminate false positives. The camera is also more capable of detecting smaller objects lying on the ground than the LiDAR, though only in ideal lighting and weather conditions. Small objects might fit between the beams of the

LiDAR and distinguishing between a small object on the ground vs the ground itself is particularly challenging [30].

Figure 2.4 Figure showing RGB camera integration into the front bumper of the vehicle. Cameras are circled in red. Figure best viewed in color.

The detection package provided information to the control system detailing where beacons were in the tractor's environment. Beacons are typically placed around valuable equipment that could be subjected to the traffic the tractor brings into the area. Strategically placing beacons around the area wished to be blacklisted from tractor movement allows continuous operation for the tractor, but also helps to ensure the safety of the designated area and prevent expensive equipment from costly damage. The MPC module is configured to automatically set to upper limit of the vehicle speed to 2.5m/s when a beacon is detected. This is to help with driver adherence to the policy for the vehicle in this form of area. This policy is to be enforced even if the tractor-trailers are not on a collision course. Having an upper speed ceiling in restricted areas helps to encourage focused navigation from the tractor driver.

2.4.2 Tractor/Trailer Model Integration

The complexity of tracking the tractor/trailer train throughout space required the development of a vehicular model that could be used in the MPC algorithm. A polygonal model was developed to be used to approximate movement of the vehicular system through space [4]. With the assumption that the tractor is globally referenced to the center of the vehicle, meaning the model assumes the world moves around under it, the polygon model could account for the tracking of the trailers attached to the tractor. This model was proven to be able to record the progression of the tractor and trailers with high fidelity; and by using this polygon model the approximate area of the multi-trailer body could be estimated for the MPC algorithm. This can then be used to prevent the modeled polygon from entering the area blacklisted by the placed beacons. The MPC levying the model produced by this work can now account for vehicle movement by using the sensor data. An example of the tractor/trailer model being used to determine movement can be seen in simulation for Figure 2.5 and physically in Figure 2.6 below. The MPC algorithm evaluates the moving system's location and ensures that if a collision is plausible, it can intervene to halt the vehicle before it occurs.

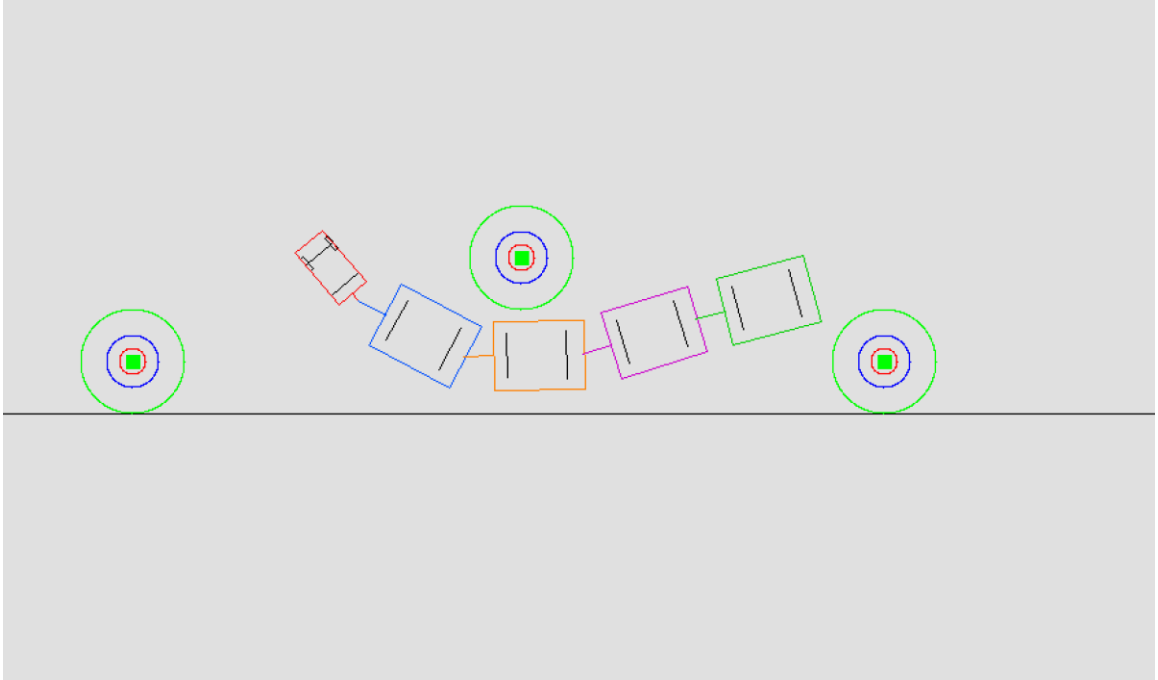

Figure 2.5 Capture depicting virtual model of a tractor-trailer system navigating beacons. This is a top down view. Figure best viewed in color.

This figure depicts a simulation of the vehicle driving between beacons. The green ring around the 3 beacons shows the area the controller tries to prevent travel into. Movement inside of the blue rings around the beacons results in the controller commanding an emergency stop to prevent collision. The long black line shows a planar axis of the 2D space.

Figure 2.6 Photo of the physical tractor-trailer system traversing around a beacon.

2.4.3 Vehicle System Expectations

It is desired for a series of beacons to be placed around expensive equipment or restricted areas, like a fence, to prevent a tractor driver from being able to accidentally enter these areas. The distance restricted around these beacons can vary but was typically tested for a 3-meter radius around the beacon. The sensor package can update object detection information at a rate of 5Hz. The MPC algorithm can then take in this information and extrapolate the vehicle model based on the number in different control actions and lookahead steps specified. If the minimum keep-away distance is breached, the MPC algorithm will force a complete stop of the system to prevent collisions.

After testing, it was determined that the loaded tractor-trailer system was only capable of slowing down stably at a rate of 1m/s^2. Rates faster than this could cause the brakes to lock resulting in the tractor skidding beyond driver/MPC control. The vehicle is capable of a maximum speed of 5m/s. The MPC control inputs consisted of four deceleration rates scaled from 0m/s^2 to 1m/s^2 depending on current speed being traveled. This is depicted below in equations (2.1 – 2.4) below. Since the most the system can decelerate is -1m/s^2, the strongest input is constrained to ensure this condition. The fractional deceleration allows for smoother speed transitions depending on the starting speed of the vehicle. A slowdown of 0.5m/s^2 when traveling at 5m/s is much smoother than when traveling at 1m/s. This scaling helps to reduce abrupt braking which can cause unpleasant rides for the driver and/or passenger.

$$a_{1out} = 0\, m/s^2 \tag{2.1}$$

$$a_{2out} = \frac{v_{current}}{5m/s} * \frac{-1}{3} m/s^2 \tag{2.2}$$

$$a_{3out} = \frac{v_{current}}{5m/s} * \frac{-2}{3} m/s^2 \tag{2.3}$$

$$a_{4out} = \begin{cases} \frac{v_{current}}{5m/s} * -1\, m/s^2, \frac{v_{current}}{5m/s} * \frac{2}{3} m/s^2 > -1\, m/s^2 \\ -1\, m/s^2,\ else \end{cases} \tag{2.4}$$

The time taken to decelerate can be determined using equation (2.5) below. The distance needed to stop is demonstrated by equation (2.6) below. The braking mechanism and introduced software latency also influence total stopping distance when a speed reduction is needed, this is characterized by $t_{latency}$ in (2.6).

$$t_{decel} = -v_{start}/a_{decel} \quad \text{(seconds)} \tag{2.5}$$

$$x_{stop} = -\frac{1}{2} * v_{start}^2/a_{decel} + v_{start} * t_{latency} \quad \text{(meters)} \tag{2.6}$$

These rulesets and understandings were used in tuning MPC so that it can work within the physical restraints set for the CAS. A design compromise of smoothness in control and effectiveness was targeted such that the CAS can work as expected while also allowing for the necessary work of the driver on the vehicle.

2.4.4 Control Communication

The pre-existing Controller Area Network (CAN) on the tractor provides the most crucial communication link between devices. CAN is a robust communication protocol that is commonly found in modern vehicle communications systems [31], [32]. The differential signal CAN uses provides superb noise protection and even allows for multiple devices to be interconnected on the same channels [33]. Both the Actuation Controller and the CAS heavily used this last feature to prevent added latency of one system reporting speed to the other. **Error! Reference source not found.** demonstrates the critical CAN components on the CAN network installed on the tractor.

The CAN network used is an extended frame format. The main difference between extended frame and standard frame is that extended frame messages have 29-bit message IDs and standard frame messages only support 11-bit message IDs [33], [34]. Message IDs help to distinguish the data being sent from all the other possible broadcasts on the bus. Each CAN message can have up to eight bytes of data, for a total of 64-bits in the data fields [33], [35]. Data larger than eight bits can be sent inside of a CAN message, but it must be split into multiple eight-bit sections. This means that every message sent on the network must have some known data scheme for each individual message such that any split data can be correctly reassembled and correlated to its significance.

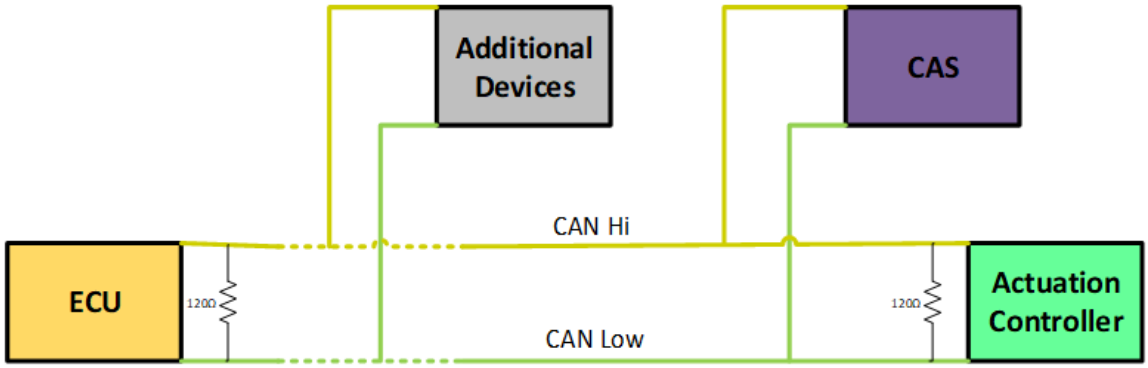

Figure 2.7 Diagram representation of vehicle's CAN network

A detailed view of how the CAN bus was used in this study is reviewed in section 4.5 below.

2.4.5 Physical Control Algorithm Integration

Since this controller is being implemented in a dynamic vehicle with real-time requirements, the CAS solution must be constrained against obtainable time deadlines and have adequate amount of time to respond to the found solution. The algorithm was developed to run on a sophisticated embedded platform. This embedded controller is capable of computing control decisions in the time constraints needed for real time control. The embedded platform is interfaced with the installed sensors and obstacle detection package, and it is also able to communicate with the actuation controller on the tractor. This close-knit integration allows for the embedded platform hosting the algorithm to be able to directly receive the required system metrics for its operation. This combination of outside data is used inside of a previously developed model representing the dynamics of the tractor and multiple subsequent trailers traversing throughout space.

The MPC algorithm is expected to return a control solution the actuation controller between updates received from the sensor package. Since updates from the

obstacle detection algorithms are received every 200ms, it is expected for the MPC algorithm to receive this data and find an optimal control solution between sensor messages. These control actions are then sent to the actuation controller to ensure the optimal control action is enforced. The MPC algorithm can run at a rate faster than it receives new obstacle data, each iteration a new vehicle state is considered and allows for quicker reaction time if the vehicle is on a collision path. During these intermediate checks between obstacle detection updates, the CAS will approximate the beacon locations and update them with live data once new obstacle locations are sent.

CHAPTER III

ALGORITHM DEVELOPMENT

3.1 MPC Structure

The MPC algorithm is designed to evaluate possible control solutions from a predefined set of control actions. These actions are the variations in speed the tractor can physically make between time steps of the MCP algorithms. The vehicle's current state, and the effect of each action are then evaluated inside of a cost function that tries to prevent collisions into areas restricted by placed beacons or front guard detections. The acceptable proximity the tractor can get within the beacons is a definable parameter. The MPC algorithm minimally limits the driver's speed such that the tractor can be allowed to travel as close to the beacon(s) as desired without entering the area around the beacon defined as not traversable. Entry into this area is classified as a collision and prevents the driver from progressing any further without a change in the driven trajectory. Since there is not a means to autonomously control tractor steering, the CAS does not control the tractor and trailer's heading.

3.2 Collision Avoidance with MPC

The vehicle model accepts parameters including speed, heading, and duration of travel when it is propagating the vehicle's possible movement in the world. A weakness in the model is when the avoidance system runs from a "cold boot" the trailers behind the tractor are in an indeterminate state. This state means that the controller is not properly

aware of the orientation of the tractor or trailers. The tractor must travel a small distance for the actual position of the trailers to realign with what the model is estimating. After this alignment period, the CAS controller should have a good understanding of the vehicle model it is controlling. By combining the information provided by the obstacle identification platform with data given with the vehicle model, it is then possible to determine if the tractor is on a possible collision path into any beacon areas.

A Limited Lookahead Controller (LLC) was developed initially and proven to be effective but was later replaced by a superior Worst Path Planning (WPP) algorithm. The LCC approach will first be discussed, and later contrasted to the WPP method. Since the MPC algorithm has a finite amount of control inputs offered to the vehicle, it is then possible to generate a set of all possible control actions. This set of actions is not only defined by the set of control actions, but also the number of steps wished to project the vehicle into the future. The length of these times steps will be covered later.

3.2.1 Limited Lookahead Control Approach

The number of potential control sequences can be found by the result of raising the size (or number of inputs) of the control set to the power of the number of lookahead steps desired. After calculating the number of possible sequences, it is possible to calculate the sequence sets, which are shown in Figure 3.2 below. Each row in this data set would then be passed with current vehicle metrics into utility function of the algorithm to find an optimal control candidate. Table 3.1 demonstrates the resulting sequence produced from a sample control input set of {0, 1, 2, 3} evaluated at three time-steps into the future. The number of control inputs and lookahead steps were chosen to simply demonstrate structure of control options.

Table 3.1 Example of Sequence Produces by Four Control Actions with Three Lookahead Steps

	1st Control Action	2nd Control Action	3rd Control Action
Sequence 1	0	0	0
Sequence 2	0	0	1
Sequence 3	0	0	2
Sequence 4	0	0	3
Sequence 5	0	1	0
. . .	. . .	. . .	. . .
Sequence 64	3	3	3

Increases to either the amount of control actions of the number of lookahead steps exponentially increase the computational complexity of checking every combination. With more lookahead steps and larger control input sequences, this table will grow exponentially larger.

The generated control sequence can directly represent a decision tree, where the node of the tree is the current state of the vehicle before the controller attempts a path optimization. A short representation of this form can be seen in Figure 3.1 below. This figure also helps visualize the exponentially growing nature of the sequence generation. Each possible path you can follow in the tree represents a sequence demonstrated in Table 3.1. Each of these paths demonstrates one of the repetitions that must be made by the controller, and as the result sequence grows so does the amount of demand that is put onto the collision avoidance controller. Parallelization of this process for improved performance will be discussed later.

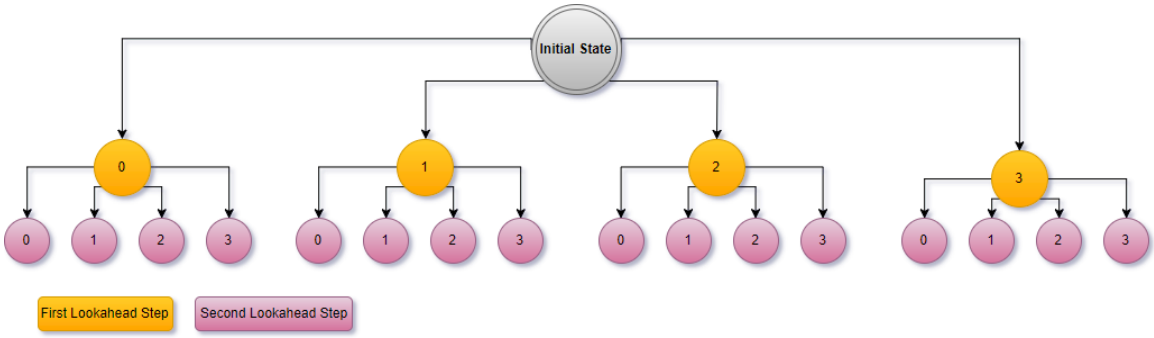

Figure 3.1 Visualization of the 2 lookahead step spanning tree. Each number here represents a unique control the MPC algorithm could enact on the system. Image is best viewed in color.

This tree shows the various path of dynamic system accounting for only one control variable. A multiple variable tree would be far more complex.

The list of control sequences along with the vehicle movement data and a list of detected obstacles are passed into the algorithm which then begins to optimize for the least limited but collision free control for the vehicle. Figure 3.2 below demonstrates a means to create Table 3.1 in code. The LLC algorithm optimizes its control sequence based on the assumption the tractor will be making the most direct path to each beacon. For each sequence, the outcome of the tractor is projected and each corner from the tractor and trailers is checked to see if they are within a restricted area as demonstrated in Figure 3.3. If the model can project the vehicle model into every lookahead step, then that sequence is accepted as being valid. If the projection causes the vehicle model to enter a blacklisted area, the controller will reject this solution. The highest control input that ensures the tractor will not enter a restricted zone is selected out of the valid potential control candidates.

```
//Method to generate the complete set of control sequences.
short size_data = 4;
short lookahead_steps = 4;
short data[size_data] = {-3, -2, -1, 0};

//InputSequence uses linear addressing by the equation of [row + (column * #lookahead
steps)]
short *InputSequence = new short[int(pow(size_data, lookahead_steps))*lookahead_steps];

int i = 0, j = 0, k = 0;
int counter = 1; // should be initialized to 1 because it is an understood start value
for my algorithm

// first two while loops control matrix addressing
while (i < lookahead_steps) {//columns
	while (j < pow(size_data, lookahead_steps)) { //rows
		while (k < size_data) {// this loop steps through our input sequence
			InputSequence[i + lookahead_steps*j] = data[k]; //assigns value from
		input values
			if (counter == pow(size_data, i)) { //counter controls how often the
		value k changes, or rather how often we parse through our input data.
				k++;
				counter = 0;
			}
			counter++;
			j++;
		}
		k = 0;
	}
	i++;
	j = 0;
}
```

Figure 3.2 Iterative method used to create control sequence table in software. Taken from C++ source code.

The LLC is the more conservative approach of the two developed, meaning that this controller impacts the control of the driver the most and potentially unnecessarily. This approach also does not fully utilize the dynamic path determination the vehicle model offers. Since we are assuming the tractor is driving a straight path into each beacon, the pulled trailers tend to straighten out during projections providing an unrealistic representation of travel when projected. This approach would also make assumptions that were physically impossible. If a beacon was located directly

perpendicular to the vehicle, but far enough away such that the vehicle was not in a regulated control zone, the controller may assume the vehicle could head directly towards beacon making an instantaneous 90° turn.

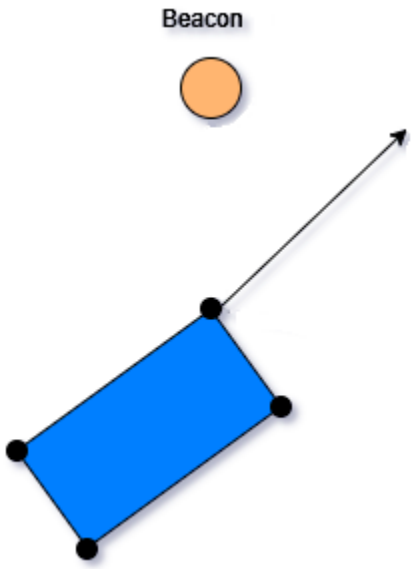

Figure 3.3 Illustration showing distance checking between the vehicle and obstacles for LLC design.

3.2.2 Worst Path Planning Approach

To rectify the shortcomings of the LLC algorithm, a means to account for changes in vehicle heading were added into the optimization function of the MPC controller. As the vehicle model is projected into the future by the algorithm, a means of determining the angle of obstacles are apart from the tractor was introduced. A check was introduced at the beginning of each timestep to see if the angle from the tractor to an obstacle was greater than the maximum amount of turning the vehicle can make in one timestep. If this angle was greater than the vehicle was capable of pivoting, the next time step was repeated in this manner until it was found that a collision was possible, or the extent of

the predictive sequence ended. The propagation described here is demonstrated in Figure 3.4. If no sequence was identified as posing the potential for a collision, no movement restriction would need to be made by the controller. If one or more possible collision paths were identified, the optimizer would identify the least restrictive movement sequence possible that would not cause the vehicle to make a collision into a restricted region.

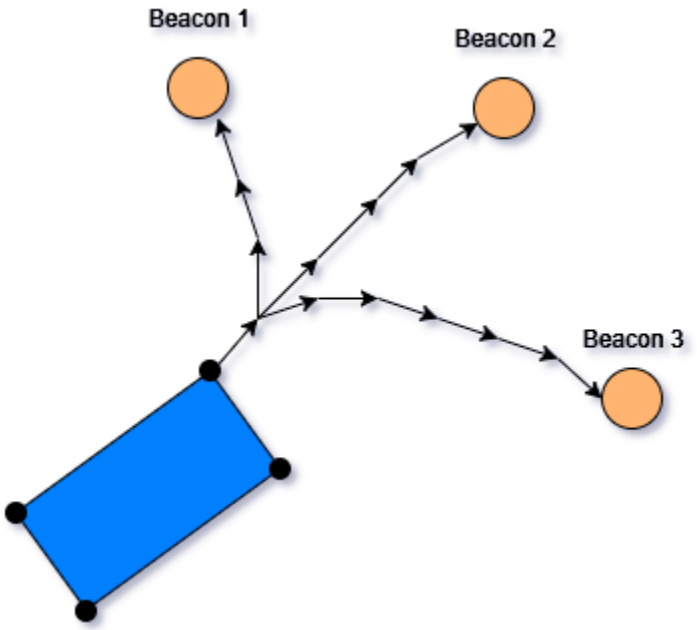

Figure 3.4 Visualization showing distance checking between the vehicle and obstacles to avoid collision accounting for the maximum expected change in heading for the WPP approach.

Each arrow represents the approximation of travel assuming a direct collision course.

3.2.3 Performance Optimization via Parallelizing

As previously touched on, the repetitious nature of the MPC algorithm has proved to be the most performance draining aspect of its implementation. The more granular the

time steps or the sheer amount of time steps being evaluated has a substantial impact on the sequence size the controller must calculate. It was identified that every lookahead check performed was comparable in structure in each iteration. This idea sparked investigation into finding a way to multitask these sequences since they are independent of each other. General Purpose Graphics Processing Units (GPGPU) were found to be a natural fit for the controller processing as they can support thousands of simultaneous running threads [36]. To port the MPC controller into the GPGPU, it was realized that the pre-generated sequence structure was becoming a bottleneck due to its structure size and how the hardware handles storage and access of such data. A method was then identified that would allow each thread to be able to independently generate their unique sequence to evaluate by just referencing its unique thread ID. This allowed the pre-generated sequence to not have to be handled by each thread of the GPGPU, which was causing a large degradation on performance due to memory access speed. The sequence generation code demonstrated in Figure 3.6 below was found to significantly improve computation over the original GPGPU implementation, which was an outstanding speed-up over a strictly CPU-processed control means as shown in Figure 3.5 below. The GPGPU implementation of this algorithm was performed in parallel of this work as a separate study.

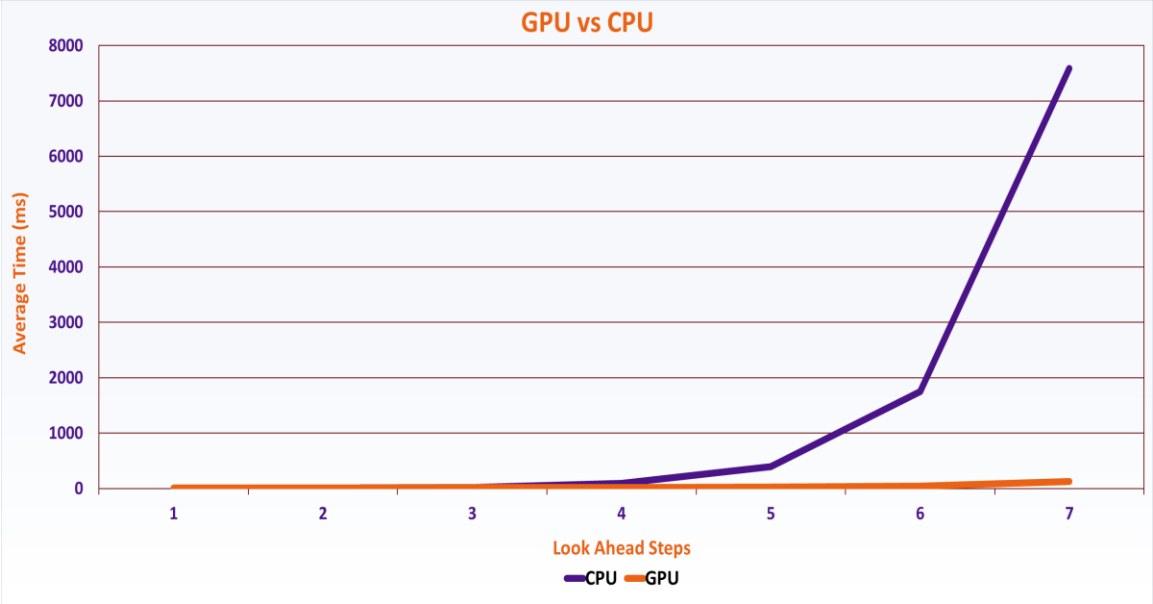

Figure 3.5 Graph showing the speedup significance of parallel MPC calculations.

```
%Unique Thread Based Sequence Generator
lookaheadsteps = 5;
acceleration_inputs = [0, -1, -2, -3];
acceleration_input_size = length(acceleration_inputs);
num_threads = (acceleration_input_size.^lookaheadsteps);
sequence=zeros(num_threads,lookaheadsteps);
for t = 1:num_threads %should be 0 to num_threads-1
   for i = 1:lookaheadsteps % should be 0 to lookaheadsteps -1
      val = floor((t-1)/(acceleration_input_size.^(lookaheadsteps-i))); %t-1 fixes start with 1 indexing
      val = 1+mod(val,acceleration_input_size);% 1+mod fixes start with 1 indexing
      sequence(t,i) = acceleration_inputs(val);
   end
end
```

Figure 3.6 Instantaneous method of generating a control sequence requiring only the sequence number and desired lookahead step.

The above code in Figure 3.6 above was taken from a Matlab method that generates the sequence table by passing in a list of all sequence numbers and every lookahead step position. The method's three most nested lines of code allow for quick retrieval of the needed control action given only two reference values; the thread ID and

the current lookahead step. This instant retrieval prevents having to traverse a memory matrix, and the lack of stored matrix greatly reduces the software's memory footprint.

CHAPTER IV

HARDWARE OVERVIEW AND INTERGRATION

4.1 Requirements

The hardware installed on the tractor is expected to be able to hold up to the rough outdoor environment the tractor will work in. The obstacle detection processor and the MPC control processor were installed in a cooled, weatherproof box that prevents exposure to the environment and helps prevent overheating. This box mounting can be seen in Figure 4.1 below, and in the hardware mounted inside shown in Figure 4.2 below. Another researcher developed a power supply board to provide the required power needed for the processing units and the installed sensors. The communication cables were routed inside of the CAS housing through sealing rubber grommets. Wireless antennas were also routed through the housing so that the processing units could be remotely accessed over a wireless network. This served as a development convenience and not impactful on core research goals.

Figure 4.1 Image showing the environment-controlled box with mounted processing hardware inside.

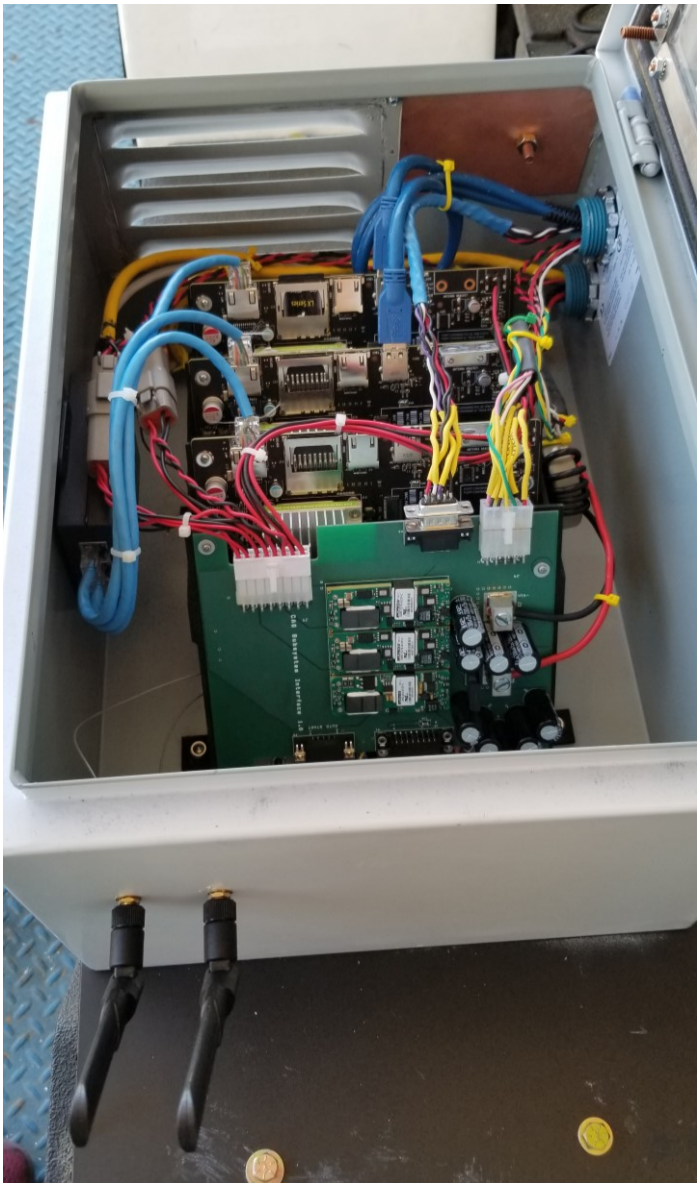

Figure 4.2 Image showing the MPC and Sensor processing units. Also shown on top of the stack is the custom power supply/conditioning board providing power to the CAS system.

4.2 Actuation/Driver Intercept Control Board

To achieve reliable system control, a dedicated microprocessor was installed, separately from the CAS, on the vehicle to control critical system components. This

control is accomplished by intercepting the vehicle throttle and by controlling an installed brake actuator. The controller brakes in addition to the driver's applied braking force and never applies more throttle than the driver is currently applying. This ensures that the vehicle behaves in a predictable manner, which is important when implementing a reactive support system such as a collision avoidance package. When an avoidance action is required, the controller will limit the amount of throttle output going to the engine as shown in Figure 4.3 below. If required, a braking force is applied to reduce the system's speed, this is shown in Figure 4.4 below. This braking is critical when attempting to avoid collision with a detected object.

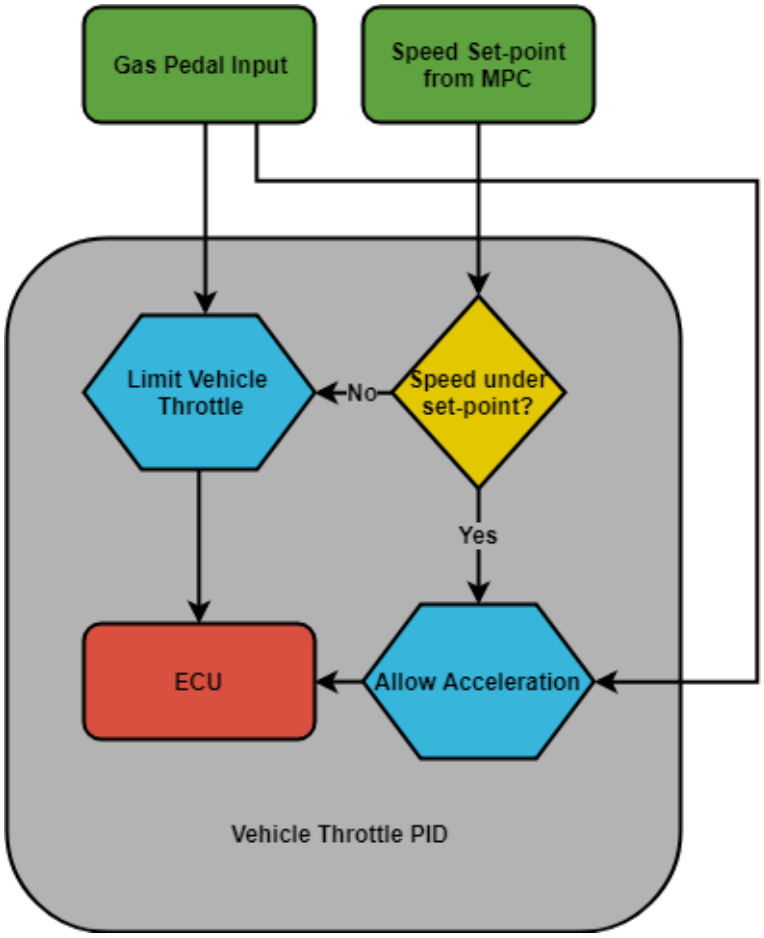

Figure 4.3 Figure depicting the vehicle throttle control logic on the actuation control board.

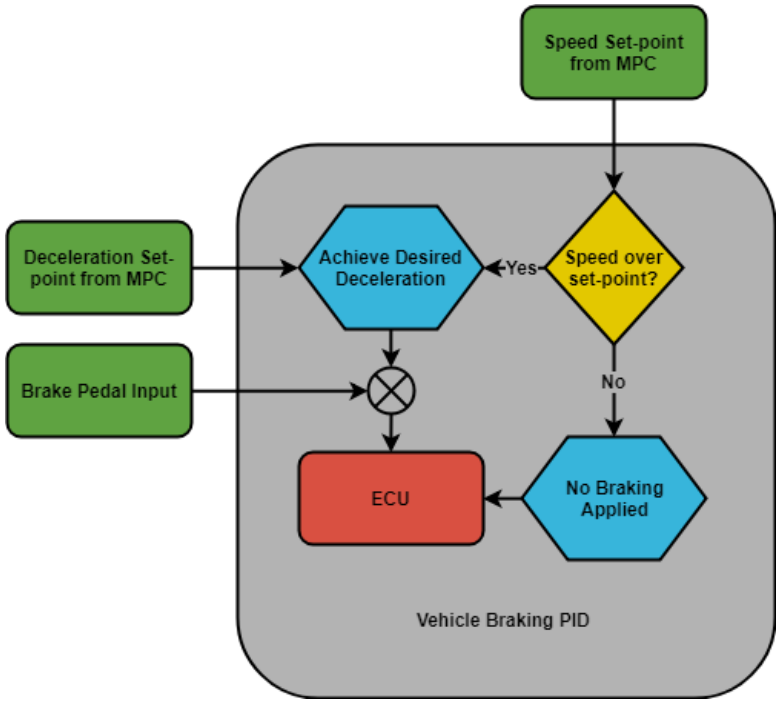

Figure 4.4 Figure depicting the vehicle braking control logic on the actuation control board

The controller is reliant on intercepting CAN messages from the tractor's ECU and the CAS to perform system actuation. The ECU provides the controller with real time system information while the CAS sends any avoidance measures the vehicle needs to make if the driver maintains the current path. Combining the real time speed of the tractor and the requested control from the CAS allows a dual, closed loop PID control system to physically control vehicle speed. These two PID loops independently control the throttle and brake output from the controller that goes to the mechanical systems.

4.2.2 Control Device and Methods

To ensure that the tractor maintains full system responsiveness, a dedicated microprocessor was utilized for system controls. This ensures that any processing latency

from the CAS does not affect the vehicles ability to perform a control action or operate properly. A Microprocessor Unit (MPU) was selected around computational needs and auxiliary devices it interfaces with. A processor with 200Mhz clock rate was selected since the device is responsible for real time systems. To best fit the computational needs of the system and the required communications/interactions for proper control an Arduino Due was selected for the onboard processor shown in Figure 4.5 and Figure 4.6 below.

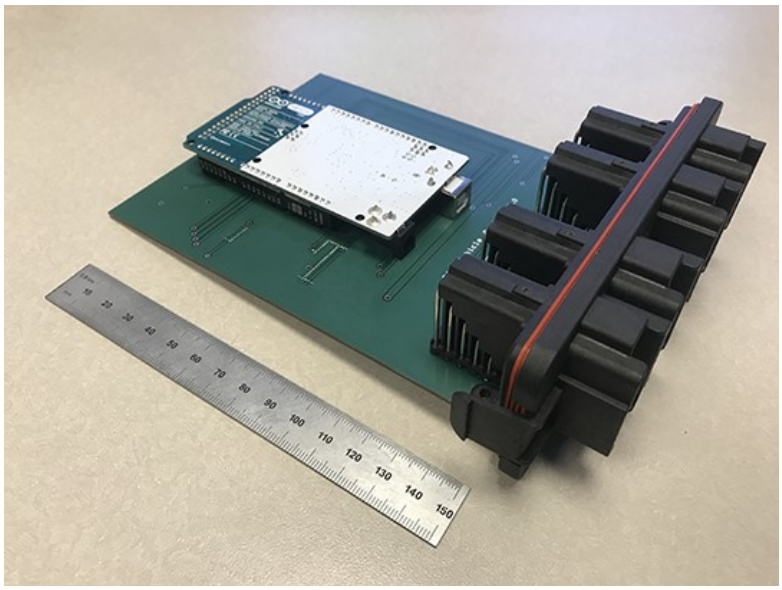

Figure 4.5 Image of the actuation interface board with Arduino Duo microcontroller mounted.

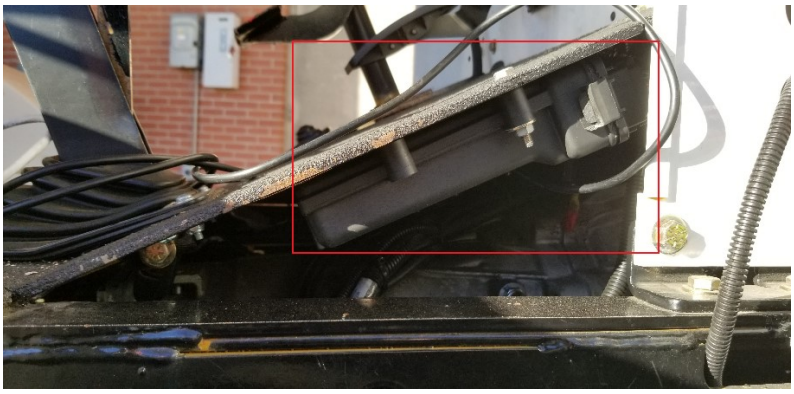

Figure 4.6 Image of the actuation control board discretely mounted under the floorboard of the cargo tractor. Mounted board demonstrated in red box. Figure best viewed in color.

The Arduino Duo was selected in part because of its CAN interface capabilities that allowed for easy integration into the vehicle's ECU. The weather proof connectors route signals both to and from the controller. Interfaces like CAN, braking, acceleration, and more are connected directly into the controller. Since it is expected for the tractor to operate in harsh weather and extreme conditions, care was put into the design of retrofitted additions to make sure that it will hold up to the demanding environments. Weatherproof housing and cables were used to ensure system reliability.

The Arduino Due has 12-bit Analog to Digital Converters (ADCs) that are used for reading driver induced throttle and braking pressure. 12-bit Pulse Width Modulation (PWM) is used to output the controller's throttle and brake signals to the vehicle. CAN communication is used to read messages from the vehicle ECU for current speed and from the CAS to get current control solutions. Additionally, the controller operates the digital gear shifter by manipulating the gear position by means of a servo that allows for

buttons to be pressed to change the vehicle from and to Park, Reverse or Drive. The digital pin interrupts on the controller detect when a button is pressed and will then respond by putting the gear servo in the corresponding position. This is demonstrated in Figure 4.7 below.

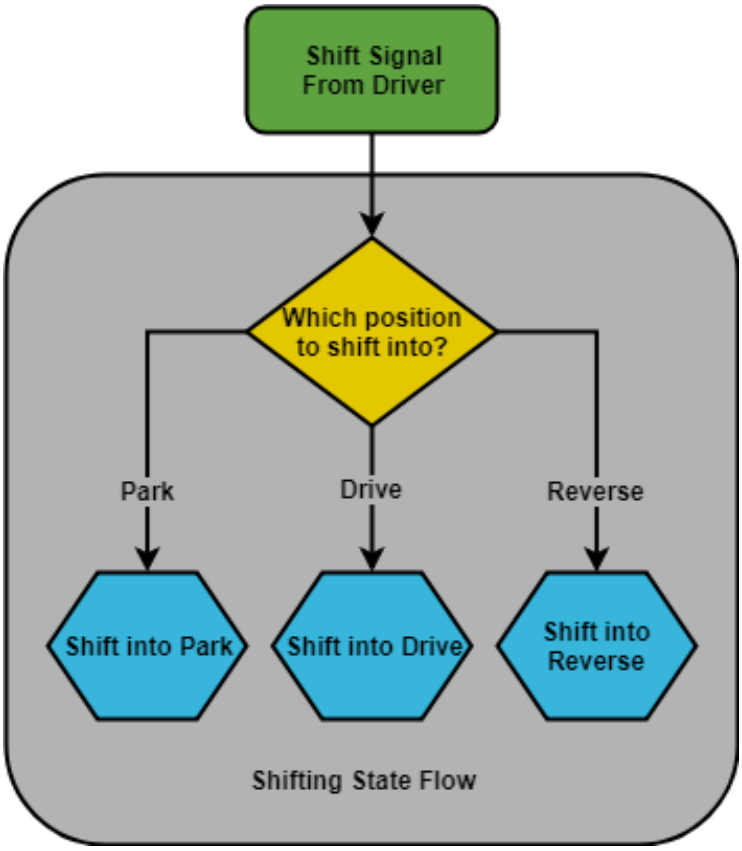

Figure 4.7 Figure showing the state transition of the gear shifter.

4.3 System Manipulation

The control actions are managed by physical systems installed on the vehicle. Both throttle and brake control outputs from the actuation controller manifest themselves as 4kHz PWM signals. These signals are operating at a resolution of 12-bits. Meaning an

output response of 0% would have a digital value of 0, while an output response of 100% would have a digital value of 4095 ($2^{12} - 1$). This resolution provides a control granularity of 0.000244, or roughly 2.5 ten-thousandths of variation between each possible control output. Additionally, a three-button array is installed on the dash of the tractor that creates a digital signal to shift the vehicle into either Park, Reverse, or Drive.

4.3.1 Brake Actuator

The brake actuator installed on the tractor creates braking pressure based on the duty cycle of the controller signal. The operating points of the braking loop are defined around the limitations of the brake actuator. Since the brake actuator only responds to a limited signal input range, the brake PID loop operates in the range of the component to provide the best control. This input signal range was measured to be 25-70% duty cycle. The brake actuator ignores inputs of less than 25% duty cycle, and the response caps at around 70%. Setting the operating range around these points allows for a finer control response. A floor of 25% duty cycle will have less latency in braking than if the range starts at 0%. Similarly, a ceiling of 70% will prevent the braking loop from overshooting the needed control setting than if operated until 100%. This would cause disruptive, coarse braking and is undesirable for a well-tuned system control.

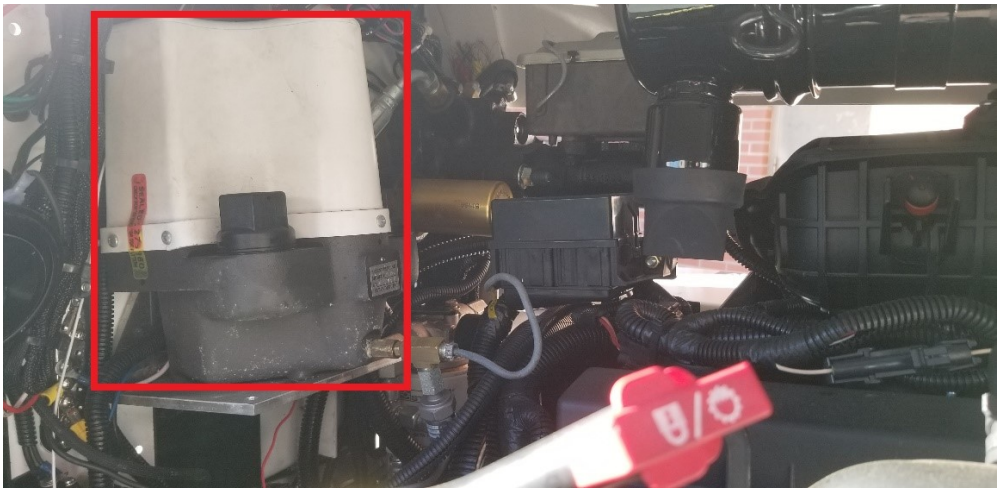

Figure 4.8 Image of the installed brake actuator installed on the ground vehicle highlighted in red. Figure best viewed in color.

4.3.2 Throttle Output

The throttle value from the gas pedal on the tractor is read as an analog value on the actuation controller's ADC in the range from 0 to 3.3 volts. The actuation controller will then either pass through or adjust the throttle value before sending the signal to the ECU. This adjustment is taken if the actuation controller is trying to achieve a speed below what the driver is requesting. The output from the controller is passed into an analog filter that will convert the digital PWM waveform to an analog signal. The MPU throttle output is ranged from 20% to 100% for similar reasons as the brake scaling. The tractor's ECU expects a minimal voltage to be read on the throttle. If the throttle output drops below this nominal value, then the throttle will lockout on the ECU and prevent acceleration until the ECU is restarted.

4.3.3 Shifting Servo

The shifter button presses allow a digital signal to be interpreted by the controller which operates on a digital interrupt. For our MPU, an interrupt is generated when the system is given an immediate action that will take place immediately over what is currently being processed. This functionality is desired for any form of user input because the system is unable to know how long the given input will be available, like how long will the button be pressed, and should handle it promptly to prevent missing any input. Each button signal is attached to an individual function that will shift the gear accordingly if the required safety checks are validated. These checks involve checking the vehicle is not moving or is in park and making sure the servo is not handling another shifting operation. This second check prevents undetermined functionality if two buttons are pressed at once.

The controller can change the gear position by sending a 50Hz PWM signal with a varying duty cycle that the servo interprets as a position to move to. This frequency is much lower than our other PWM frequencies because with servo position control, less granularity is needed to achieve a comparable level of precision. Most servos follow a loose standard of this frequency which is common in servo control.

The gear shifter is mechanically attached to this shifter and will move corresponding with any servo movement. At 50Hz, the PWM signal of the servo driver has a pulse width of 20ms. To achieve the park position, a high period of the pulse of 1.415ms to be written, providing a duty cycle of 7%. Reverse gear requires a high period of 1600ms giving a duty cycle of 8%, and the drive position requires a high period of 1840ms giving a duty cycle of 9.2%. These duty cycles actuate the servo to turn to a

position between far left and far right in a 180° range. The values were determined by figuring out where the servo was in each of the corresponding gear locations.

4.4 PID Control

The characteristics of both throttle and brake response were recorded and an initial transfer function was estimated for each using Matlab. PID values were extracted from these transfer functions are baselines for the control loops. The PID values were then adjusted by hand to the exact performance desired. This last step was necessary because the tuned data set could have uncompensated error when modeling.

4.4.1 Throttle Control

The throttle passthrough performed by the controller allows a PID loop to maintain a maximum set speed defined by the CAS. While the speed is being throttled, the driver can achieve speeds up to but not exceeding this capped limit. If the vehicle's speed meets the set control speed the output of the PID will stop increasing and maintain the speed set point. If the CAS sets the vehicle speed lower than the driver input, the throttle PID will back down the throttle output until the required speed is met.

Since the throttle is intercepted by the actuation controller, the throttle pedal position is not always representative of the throttle output. If a speed set point is being held by the throttle PID loop, then any throttle applied to the pedal that is higher than the PID set point will be ignored. The system will always output the lesser of the two possibilities; this means that the actuation never actively applies acceleration to the engine. The driver will always have to request movement over creep (the natural idle speed of the vehicle in gear) if the tractor is in a driving gear. This ensures the system

performs in a predictable manner and serves as a safety against unwanted accelerations. If the CAS allows a speed higher than the current set point, the PID controller allows the tractor to accelerate up to the new speed in a smooth manner when and if the driver applies additional throttle.

4.4.2 Brake Control

The brake PID loop is based on achieving a determined deceleration rate. This allows for reducing the system speed in a smooth, refined manner that also meets the required CAS control responses. This is important because the CAS intelligently determines the maximum allowable driving speed the vehicle can travel in each time step, and the actuation controller must be able to meet the speeds requested to prevent a collision. The CAS accounts for the system's physical braking limitations and requests achievable goals for the braking PID loop to meet. The refinement of the PID calibrations allow for smooth speed transitions even when the tractor is loaded with dollies.

In contrast to how the throttle output is intercepted by the actuation controller, any brake force produced by the automated system is additive to driver braking. This ensures that braking is always achievable by the driver. Assistive braking has been added through MPU software that detects when the driver applies a braking pressure and the mapping of the brake actuator output to that of the driver. This serves as a driver convenience allowing for the driver to be able to slow or stop the system with less force applied to the brake pedal.

4.5 Communication

The actuation controller code has a developed console interpreter that allows for system messages and commands to be sent manually by connecting a laptop to the controller. This functionality allows for inducing controller speed set points and dynamically calibrating the PID values of the PID control loops for tuning purposes. Also, a desired speed set point and/or deceleration rate could be configured for evaluating system performance to given inputs. This communication also allowed for recording vehicle data that was used in tuning both PID control loops.

4.5.1 Electronic Control Unit

The ECU device was previously a black box that was examined, and crucial CAN messages reported from it were identified. Specifically, two messages were found to be especially relevant to the CAS and actuation controller. One message tells the current state of the gear position, which is used by the CAS and actuation controller. This message ID is 0x18FF0203, and byte 5 is used to determine the position via an unsigned short integer. A value of 8 means the vehicle is in park, 6 means the vehicle is in reverse and a value of 3 means the vehicle is in drive. The other message utilizes a data field sent that combined with a ratio provides the current speed of the vehicle. Its message ID is 0x18FF0503 and bytes 2 and 3 of the data portion of the message hold the value speed is extracted from. Byte 2 of the data set is the Most Significant Byte (MSB) and byte 3 is the Least Significant Byte (LSB). The notation of MSB and LSB allows the developer to properly combine split data packets in the correct order; without such notation it would be difficult to know how to correctly reassemble the received data.

4.5.2 Collision Avoidance System

The CAS reads the messages described in section 4.5.1 above to achieve its desired functionality. The gear position message allows for the system to know whether it has been shifted in park after a forced stop. Once this gear shift has taken place, the CAS will be temporarily disabled so the driver can get the vehicle out of the undesired position. Also, the vehicle speed is determined by sampling the CAN bus, this allows the CAS to know what speed this system is currently traveling so it can effectively use MPC to find the least restrictive speed to prevent collisions.

A CAN message demonstrated by Table 4.1 is maintained on the CAS that is broadcast periodically intended for both the actuation controller and the Status Indication Module (SIM) discussed in section 4.5.3 below. This message provides the actuation controller with the current speed governance required and current acceleration of the system. Also, this message indicates to the SIM what mode of operation the CAS is in, which allows for vehicle state indication for the driver.

Table 4.1 CAS Managed CAN Message

CAS System Information Message		
Message ID: 0x1234h		
	MSb….	….LSb
Byte[0]–[07–04][03–00]	CAS Status Encoding	
Byte[1]–[15–12][11–08]	CAS Disabled Counter Range	
Byte[2]–[23–20][19–16]	CAS Disabled Counter	
Byte[3]–[31–28][27–24]	Requested Speed 1/100th Place	Requested Speed 1/10th Place
Byte[4]–[39–36][35–32]	Requested Speed 1/1000th Place	Requested Speed Ones place
Byte[5]–[47–44][43–40]	Acceleration 1/100th Place	Acceleration 1/10th Place
Byte[6]–[55–52][51–48]	Acceleration Ones Place	Acceleration Sign (+/-)
Byte[7]–[63–60][59–56]	Max General Speed Limit	

4.5.3 Status Indication Module

An auxiliary package was added to the tractor to help provide the driver with additional information such as when the vehicle speed is being limited, or what caused a full stop to the vehicle. A four digit seven-segment display was included so that the driver can more easily determine the speed of the vehicle. Inter-integrated Circuit Protocol (I2C) on the controller is used to update a 7-segment display that showed the vehicles current speed. This display package, referred to as the SIM, is controlled by a Teensy 3.6 MPU that is attached to the tractor CAN bus. With this connection, the MPU can show the driver the current speed in m/s along with illuminating an addressable LED strips that indicate CAS status to the driver. Additionally, up to two more strips are supported by the SIM that can be optionally installed for observer benefit to see the real time reactions being made by the CAS. The SIM is shown in Figure 4.9 below.

Figure 4.9 SIM module housing the status indicating LEDs and the 4-digit display used to display tractor speed to the driver. LED colors explained in Table 4.2Table 4.2 below.

The left most 3 LEDs indicate the gate area status of the tractor. These colors are visually demonstrated in Table 4.2 below. A solid blue color is shown when no beacon is detected. These lights will flash blue when a beacon is detected, and the tractor speed is being capped at 2.5m/s. The middle 3 LEDs illuminate solid yellow when the CAS is restricting the vehicle speed. Finally, the last section shows what condition has brought the tractor to a stop by illuminating in red. If the lights are solid, the driver will know that a front guard detection caused the vehicle to be brought to a stop. The lights will flash if beacon proximity has brought the tractor to a stop. All 3 sections show independently of each other meaning that all 3 sections can provide feedback simultaneously.

Table 4.2 Normal Operation Light Sequences

Not Limited	(Solid)		
Global Speed Limited	(Flashing)		
Tractor Speed Limited		(Solid)	
Front Guard Stop			(Solid)
Beacon Stop			(Flashing)

The 9-pixel LED strip is split into 3 indication sections. Each 3 sections can show the driver information simultaneously. This table is best viewed in color.

CHAPTER V

CONCLUSION AND FUTURE WORKS

5.1 Summary

The CAS package was able to be successfully installed on the vehicle. After validating that each component of the system (CAS, Sensor Package, Actuation Controller, Communication, etc.) functioned as expected, testing of the physical vehicle began. Simulation results, as discussed in section 5.2 below, had already proved the viability of the MPC algorithm. The fidelity of the tractor-trailer model was verified to be within acceptable error (0.3m) of the physical train. It was observed that the furthest dolly back in the simulated system had the most error when compared to the physical system.

5.2 Simulation Results

Both approaches, LLC and WPP, were found to prevent collisions in simulation. The LLC method was more conservative than the WPP approach by nature, which is why development of the WPP approach was done. The simulation allowed a drive profile to be loaded which contained data simulating information that would be read from the vehicle moving through the perceived environment. Obstacles could be loaded into this world as desired, which allowed for watching the controller try to control the vehicle as in a real world in a physical implementation. The deceleration rates were determined to be initially unsatisfactory, being far too instantaneous for a real vehicle representation.

These control inputs were modified in simulation and seemed to improve the fluidity of the vehicle control. This might require further calibration in real world implementation.

5.3 Real World Results

A series of 3 tests were used to evaluate the performance of the collective CAS system. The first test involves driving the vehicle directly towards a stationary beacon with full throttle applied to vehicle. For the second test, a non-beacon object was rolled in front of the vehicle at a 10m distance while traveling at full speed.

5.3.1 Beacon Avoidance

0 below lists the measured stopping distance the tractor stopped away from a stationary beacon running at 2.5m/s on approach. A target distance of 1 m was set for the vehicle to stay away from. The bulk of the error recorded for these tests are contributed to the latency between the CAS calling for a full stop and the control action being executed. Since the keep away distance from the beacon is short, the average error seems large compared to the results This error distance has trended to be comparable with tests set for further distances to keep away from the beacon. It should be noted that these tests proved the system capable of arresting the vehicle 100% of the time before the beacon would have been struct.

Table 5.1 Table showing the results of the beacon avoidance test.

Test #	Distance to Beacon [m]
1	0.61
2	0.59
3	0.60
4	0.62
5	0.56
6	0.72
7	0.64
8	0.56
9	0.59
10	0.60
11	0.66
12	0.63
13	0.67
14	0.68
15	0.52
16	0.66
17	0.57
18	0.63
19	0.59
20	0.54

The average stopping distance was found to 0.61m away from the beacon, with a standard deviation of 0.4915. The average error was found to be 0.39m.

5.3.2 Moving Object Avoidance

This test is designed to evaluate performance of the system when traveling at full speed of 5m/s. With this test, a large Styrofoam block was used on a pull dolly to bring the object into the field of view of the front guard system at a desired point. The dolly and block were pulled into the field of view of the tractor when crossing a 10m away marker. The target keep-away distance was set for 3 meters away from the object. The results of this test are shown in Table 5.2 below. Again, the CAS system was able to prevent collisions with a 100% success rate. In this test, the average error is found to be

comparable to the error found in section 5.3.1 above. It is suspected that the error is smaller due to the system braking for a longer period resulting in finer control execution.

Table 5.2 Table showing the results of the moving object test.

Test #	Distance to Beacon [m]
1	6.81
2	6.81
3	6.66
4	6.69
5	6.62
6	6.65
7	6.76
8	6.80
9	6.83
10	6.81
11	6.76
12	6.75
13	6.66
14	6.74
15	6.83
16	6.70
17	6.78
18	6.74
19	6.74
20	6.73

The average stopping distance was found to 6.747m away from the beacon, with a standard deviation of 0.064. The average error was found to be 0.253m.

5.4 Potential Future Work

An adaptation of the original LLC approach was recognized as another potential control method. This method included the speed control inputs as previously desired but was also capable of generating a unique complete set of speed and angle deviations. This fined set, while far more computationally intensive, might be able to provide more accurate collision detections as the angle deviations are more representative of the physical movement of the tractor. The approach of generating this sequence is described in Figure 5.1. In the sequence, each control input is a pair of both a speed change and a change in angle. The decision tree created by this method is exponentially larger in size than the original LLC approach. The size of this sequence set was determined to be the size of the speed controls times the size of possible heading changes all raised to the power of desired lookahead steps. While a promising test candidate, the implementation of it might prove to be challenging due to the number of threads needed to execute. It is expected that the GPGPU integration described will make development of such a system possible due to its high threaded nature. Also, the method described in Figure 5.1 supports the unique sequence per thread generation described in the parallelization portion of this work.

```
%Unique Thread Based Sequence Generator that Accounts for Heading Changes
lookaheadsteps = 3;
acceleration_inputs = [0, -1, -2, -3];
heading_inputs = [-5, 0, 5];

acceleration_input_size = length(acceleration_inputs);
heading_input_size = length(heading_inputs);

num_threads = (heading_input_size*acceleration_input_size)^lookaheadsteps;
sequence = cell(num_threads,lookaheadsteps);

for t = 1:num_threads
   for i = 1:lookaheadsteps
      val = floor((t-1)/(heading_input_size.^(lookaheadsteps) * acceleration_input_size.^(lookaheadsteps
- i)));
      val = 1+mod(val,acceleration_input_size);
      temp1 = acceleration_inputs(val);

      val = floor((t-1)/(heading_input_size.^(lookaheadsteps - i)));
      val = 1 + mod(val,heading_input_size);
      temp2 = heading_inputs(val);

      sequence(t,i) = [temp1;temp2];
   end
end
```

Figure 5.1 Multi-variable sequence generator capable of considering speed changes and changes in heading.

Code derived in Matlab that produces table of arrays representing the output of any possible initial conditions passed into the generator. This method is significantly stronger than the method described in Figure 3.6 due to this method allowing for control of 2 independent control actions. It should be noted that executing the method would be exponentially more demanding on computing resources that methods utilized for this work.

REFERENCES

[1] J. Liu, J. L. Stein, P. Jayakumar, and T. Ersal, "A Multi-Stage Optimization Formulation for MPC-Base Obstacle Avoidance in Autonomous Vehicles using Lidar Sensor," *Proc. ASME 2014 Dyn. Syst. Control Conf.*, no. July, pp. 1–10, 2014.

[2] E. Kayacan, H. Ramon, and W. Saeys, "Robust Trajectory Tracking Error-Based Model Predictive Control for Unmanned Ground Vehicles," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 2, pp. 806–814, 2016.

[3] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, "LiDAR and Camera Detection Fusion in a Real-Time Industrial Multi-Sensor Collision Avoidance System," *Electronics*, vol. 7, no. 6, p. 84, May 2018.

[4] Y. Liu *et al.*, "Development of A Dynamic Modeling Framework to Predict Instantaneous Status of Towing Vehicle Systems," in *WCX™ 17: SAE World Congress Experience*, 2017.

[5] J. Liu, P. Jayakumar, J. Overholt, J. L. Stein, and T. Ersal, "The role of model fidelity in model predictive control based hazard avoidance in unmanned ground vehicles using LIDAR sensors," *Proc. 2013 Dyn. Syst. Control Conf.*, pp. 1–10, 2013.

[6] D. Alejo, J. A. Cobano, G. Heredia, and A. Ollero, "A Reactive Method for Collision Avoidance in Industrial Environments," *J. Intell. Robot. Syst. Theory Appl.*, vol. 84, no. 1–4, pp. 745–758, 2016.

[7] B. Morris, A. Doshi, and M. Trivedi, "Lane change intent prediction for driver assistance: On-road design and evaluation," *IEEE Intell. Veh. Symp. Proc.*, no. Iv, pp. 895–901, 2011.

[8] J. Storms, K. Chen, and D. Tilbury, "A shared control method for obstacle avoidance with mobile robots and its interaction with communication delay," *Int. J. Rob. Res.*, vol. 36, no. 5–7, pp. 820–839, Jun. 2017.

[9] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara, "Obstacle avoidance and path planning for humanoid robots using stereo vision," *IEEE Int. Conf. Robot. Autom. 2004. Proceedings. ICRA '04. 2004*, vol. 1, pp. 592–597, 2004.

[10] H. Chen, K. Chang, and C. S. Agate, "UAV path planning with tangent-plus-lyapunov vector field guidance and obstacle avoidance," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, no. 2, pp. 840–856, 2013.

[11] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1599–1616, 2012.

[12] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 338–345, 2012.

[13] F. Oldewurtel *et al.*, "Use of model predictive control and weather forecasts for energy efficient building climate control," *Energy Build.*, vol. 45, pp. 15–27, 2012.

[14] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves, "Model Predictive Control for the Operation of Building Cooling Systems," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 3, pp. 796–803, 2012.

[15] J. Rodriguez *et al.*, "State of the Art of Finite Control Set Model Predictive Control in Power Electronics," *IEEE Trans. Ind. Informatics*, vol. 9, no. 2, pp. 1003–1016, 2013.

[16] Y. Zhang, W. Xie, Z. Li, and Y. Zhang, "Model predictive direct power control of a PWM rectifier with duty cycle optimization," *IEEE Trans. Power Electron.*, vol. 28, no. 11, pp. 5343–5351, 2013.

[17] N. Sockeel, J. Shi, M. Shahverdi, and M. Mazzola, "Sensitivity Analysis of the Vehicle Model Mass for Model Predictive Control Based Power Management System of a Plug-in Hybrid Electric Vehicle," in *2018 IEEE Transportation Electrification Conference and Expo (ITEC)*, 2018, pp. 767–772.

[18] T. Wang, H. Gao, and J. Qiu, "A Combined Adaptive Neural Network and Nonlinear Model Predictive Control for Multirate Networked Industrial Process Control," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 27, no. 2, pp. 416–425, 2015.

[19] R. Zhang, A. Xue, and F. Gao, "Temperature control of industrial coke furnace using novel state space model predictive control," *IEEE Trans. Ind. Informatics*, vol. 10, no. 4, pp. 2084–2092, 2014.

[20] S. Abdelwahed, J. Bai, R. Su, and N. Kandasamy, "On the application of predictive control techniques for adaptive performance management of computing systems," *IEEE Trans. Netw. Serv. Manag.*, vol. 6, no. 4, pp. 212–225, Dec. 2009.

[21] J. V Frasch *et al.*, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," *Control Conf. (ECC), 2013 Eur.*, pp. 4136–4141, 2013.

[22] C. Shen, Y. Shi, and B. Buckham, "Integrated path planning and tracking control of an AUV: A unified receding horizon optimization approach," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 3, pp. 1163–1173, 2017.

[23] R. Amgai, J. Shi, and S. Abdelwahed, "An integrated lookahead control-based adaptive supervisory framework for autonomic power system applications," *Int. J. Electr. Power Energy Syst.*, vol. 63, pp. 824–835, 2014.

[24] N. Sockeel, J. Shi, M. Shahverdi, and M. Mazzola, "Pareto Front Analysis of the Objective Function in Model Predictive Control Based Power Management System of a Plug-in Hybrid Electric Vehicle," in *2018 IEEE Transportation Electrification Conference and Expo (ITEC)*, 2018, pp. 971–976.

[25] 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 *Volume 3: Multiagent Network Systems; Natural Gas and Heat Exchangers; Path Planning and Motion Control; Powertrain Systems; Rehab Robotics; Robot Manipulators; Rollover Prevention (AVS); Sensors and Actuators; Time Delay Systems; Tracking Control Systems*, 2015, p. V003T44A003.

[26] M. Jalali, A. Khajepour, S. ken Chen, and B. Litkouhi, "Integrated stability and traction control for electric vehicles using model predictive control," *Control Eng. Pract.*, vol. 54, pp. 256–266, 2016.

[27] M. Nauman and A. Hasan, "Efficient implicit model-predictive control of a three-phase inverter with an output LC filter," *IEEE Trans. Power Electron.*, vol. 31, no. 9, pp. 6075–6078, 2016.

[28] V. Turri, B. Besselink, and K. H. Johansson, "Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 1, pp. 12–28, Jan. 2017.

[29] A. Afram and F. Janabi-Sharifi, "Theory and applications of HVAC control systems - A review of model predictive control (MPC)," *Build. Environ.*, vol. 72, pp. 343–355, 2014.

[30] X. Meng, N. Currit, and K. Zhao, "Ground filtering algorithms for airborne LiDAR data: A review of critical issues," *Remote Sens.*, vol. 2, no. 3, pp. 833–860, 2010.

[31] A. Burns and R. I. Davis, "Mixed criticality on controller area network," *Proc. - Euromicro Conf. Real-Time Syst.*, pp. 125–134, 2013.

[32] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol*. New York, NY: Springer New York, 2012.

[33] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle Applications of Controller Area Network," in *Handbook of Networked and Embedded Control Systems*, D. Hristu-Varsakelis and W. S. Levine, Eds. Boston, MA: Birkhäuser Boston, 2005, pp. 741–765.

[34] H. Chen and J. Tian, "Research on the Controller Area Network," *2009 Int. Conf. Netw. Digit. Soc.*, no. 1995, pp. 251–254, 2009.

[35] H. Chen and J. Tian, "Research on the Controller Area Network," *2009 Int. Conf. Netw. Digit. Soc.*, no. 1995, pp. 251–254, 2009.

[36] A. Jog *et al.*, "OWL: Cooperative Thread Array Aware Scheduling Techniques for Improving GPGPU Performance," *Proc. eighteenth Int. Conf. Archit. Support Program. Lang. Oper. Syst. - ASPLOS '13*, p. 395, 2013.