April 2018

Autonomous Driving System and Power Transmission on 1/10 RC Car

Junbo Chen
Worcester Polytechnic Institute

Trung-Nghia Ngo Huynh
Worcester Polytechnic Institute

Zhizhen Wu
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

SELF-DRIVING ON 1/10 RACER CAR

A Major Qualifying Project Report

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

in

Electrical and Computer Engineering

Mechanical Engineering

Robotics Engineering

Submitted by

Junbo Chen
Robotic Engineering and Computer Science

Andrew Huynh
Robotics Engineering and Computer and Electrical Engineering

Zitong Jiang
Mechanical Engineering

Zhizhen Wu
Robotic Engineering and Computer Science

Date

[04/26/2018]

Approved

Professor **Jie Fu**, Major Advisor

Professor **Carlo Pinciroli**, Co-Advisor

Professor **Xinming Huang**, Co-Advisor

Professor **Zhi Li**, Co-Advisor

Abstract

To study the autonomous cars which have become an interesting topic among researchers, this project aims to implement a system that combines adaptive cruise control (ACC), trajectory generation, and trajectory tracking controller on a radio control car. To ensure high safety and performance, we implemented Control Lyapunov function (CLF) combined with control barrier function to achieve ACC. The implementation of ACC demonstrates that the car is able to drive at desired speed and keep a provably safe distance from the front car. The system uses minimum-jerk trajectory generation between waypoints, using cubic spline interpolation to generate a feasible trajectory and keep track of the desired trajectory. Trajectory tracking controller is constructed using CLF and input saturation constraints. It can keep track of the trajectory while still guaranteeing the converges. In order to help the car driving smoothly below- and above-average speed, a half-toroidal continuously variable transmission (CVT) is built and attached to the central shaft. The implementation shows that the car has wider range of speed and smoother acceleration in safe conditions. With the success of this project, future MQP teams will be able to test their own trajectory tracking controllers, speed control algorithms, and localization methods.

Acknowledgement

Table of Contents

Table of Figures

1 Introduction

Twenty years ago, robotic cars that can autonomously function and communicate with humans were something coming from fantasy novels. Even ten years ago, driverless cars still only appeared on movies. A lot of people still joke about flying cars being commercially employed in 2017 just like the movie "Back to the Future." On March 2004, the Defense Advanced Research Projects Agency (DARPA) held a Grand Challenge to find any driverless car that could finish a 150-mile course by itself [1]. Even though none of the participating cars finished the challenge, this had sparked an interest in autonomous cars for researchers and hobbyists. One year later, in October 2005, in the second Grand Challenge, five driverless cars successfully completed a 132-mile course, marking the first historical milestone in autonomous vehicles [2]. Years after that, research topics and developments related to the autonomous vehicles rode on a rising wave with a lot of vehicles making the roads for test trials. Now, in 2017, autonomous cars have become an expected reality. The Global Atlas of Autonomous Vehicles in Cities, a joint effort between Bloomberg Philanthropies and the Aspen Institute, shows that 53 cities across the globe are hosting tests on autonomous vehicles or thinking about hosting autonomous vehicles, as of October 2017 [3]. Two thirds of that have been actually piloting autonomous vehicles among commercial cars. The major automotive companies have committed to creating fully autonomous vehicles and planning on releasing them as soon as possible, with Tesla in the aiming for 2018 [4]. A conservative prediction by David Galland, from Gallet/Galland Research, stated that 10 million self-driving cars will hit the road by 2020 [5], to show how sharp the rising trend of the autonomous vehicles really is.

The autonomous vehicle from this project was motivated by the Autonomous Race Cars built by the engineering teams from University of Pennsylvania and Massachusetts Institute of Technology. Their idea was to create a racing competition similar to F1, but for autonomous race cars one tenth the size of their counterparts [6]. This competition would inspire more college students and professors to build and develop more advanced race cars to compete for the best.

One other motivation for this project was looking into the behaviors of the autonomous vehicles by researching the algorithms, mechanics, and operating systems to build an autonomous car and actually building one. The public fear them to be dangerous and unreliable [7]. The team wishes to understand where the causes come from and why they are so.

One last motivation came from the fact that most projects and researches on autonomous vehicles have been kept private and undisclosed by the government and automotive companies. The team

wished to share the software as well as hardware configurations of the car to fellow students from WPI or other colleges and other interested parties so that any people without proper facilities can study and research autonomous vehicles.

This project aims to finish building an autonomous vehicle with basic functionalities; to understand the behaviors of the car; to study and apply the latest navigation, movement speed constraint, and localization algorithms; to test different combinations of various algorithms to make the car operate smoothly and safely; and to compare against other race cars to modify and adjust the car to have it on the same pace with the current developments from other colleges.

2 Background

2.1 Autonomous Vehicles

2.1.1 Autonomous Vehicle

The way that people drive on the roads will soon change in a drastic way. So far, the term "driver" refers to the person sitting behind the wheel and actually controlling the vehicle. But soon the answer to the question "Who drives your car?" will be the car itself as artificial intelligence (or AI) and robotics components are growing popular and finding themselves into the automotive industries.

Autonomous vehicles are those that require almost to no constant inputs and attention from the human drivers. They depend on various types of sensors, such as visual, motion, and orientation sensors, to scan for the surroundings as inputs; software program to process those inputs and decide a responsive course of actions; and hardware actuators and systems to execute the decisions. Technically, a finished product—a fully autonomous vehicle—can, without human intervention, navigate from the starting location to the destination on the roadways that are not modified for robotics technologies.

Already there are no vehicles with full autonomy yet but only "semi-autonomy." They range from brake assistance, such as gradual brake and sudden stop; from lane assistance, such as driving in-lane; to headlight assistance, such as automatic dimming or moving the headlights out of views of the cars from opposite direction. Major automotive companies are investing resources and capitals on fully autonomous vehicles that can drive safely. They are aiming to showcase their products to the markets in the near future, even as soon as 2018 [8].

From the perspective of the consumers, the expectations and receptions do not seem pessimistic. From a survey of Open Roboethics Institute, most people think autonomous vehicles will help reduce traffic congestion (81.5%) and accidents (64.2%) [9]. This comes as expected as an autonomous vehicle's AI should be able to determine the most logical and efficient actions on the roads all the time. Also, nearly half the respondents to the survey say that they will pay $3,000 or more for a fully autonomous version of the same car. This shows that the consumers place the values from autonomous cars in high regards and expect a lot from them.

2.1.2 Autonomous 1/10 Race Cars

University of Pennsylvania (or UPenn) wants to inspire the advancement of robotics course in colleges. That was why they promoted the F1/10 racing competition, resembling the renowned Formula One racing competition. The differences between the two competitions are that the F1/10 participating cars are one tenth the size of their counterparts, and that they are fully autonomous. The commonly suggested car from UPenn is the Traxxas Rally Radio Control Car, which is also similar to the one used for this project, Traxxas. UPenn includes the lectures and the suggested software and hardware configurations for anyone interested in participating or simply fascinated in robotics.

2.2 Continuously Variable Transmission

Robotics products capable of moving, in general, accelerate, decelerate, and stop by changing the continuous signal that the program sends to the motor through the hardware systems that control all the intended executions of the program. The Racer car is not an exception. The speed control performs well at high speed, but suffers when it comes to low speed. The car's motor has a built-in minimum speed limit, making it impossible for the program to execute low speed acceleration. The result of this is the hard jerking when the car starts running forward [10].

2.2.1 Conventional Continuously Variable Transmission

One of the solutions that can make the car start out slowly and gradually ram up the speed is to install a half-toroidal continuously variable transmission (or CVT) after the motor output to control the output speed through hardware constraints.

As its name implies, the CVT shifts continuously in a stepless manner, resulting in the following benefits: a reduction of power consumption because the motor can operate under the most efficient conditions; smooth power delivery without any shift shock; and powerful acceleration because the motor's power is delivered continuously with little loss of driving force during ratio changes.
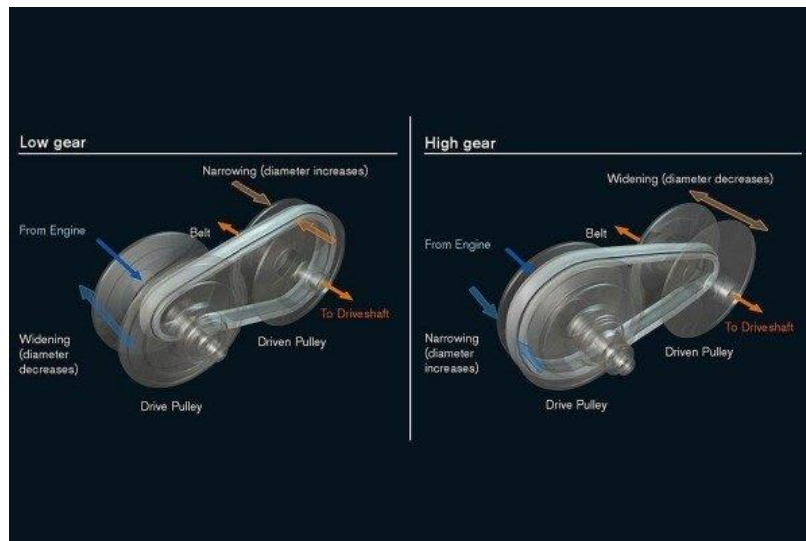
Figure 1: A Conventional Continuously Variable Transmission

A CVT, in general, differ from traditional automatic transmission in they it does not have gears that provide steps between low- and high-speed oration. The gears are replaced by variable-size, cone-shaped pulleys connected by a steel or composite drive belt. The flexible belt runs in a groove formed

between the insides of each pulley. One pulley is connected to the motor while the other is connected to the drive shaft. The diameter of each pulley is controlled by the software program, allowing the belt to ride lower or higher along the walls of the belt, thereby changing the gear ratio. Similar to how a bicycle chain moving up and down the gears, when the diameter of the drive pulley decreases and driven pulley increases, the transmission is in low gears to provide initial acceleration, and vice versa, when the diameter of the drive pulley increases and driven pulley decreases, the transmission is in high gears to maximize power usage. The nature of the CVT can increase the efficiency of power consumption by delivering an infinite number of smooth transitions from low to high.

2.2.2 Half-Toroidal Continuously Transmission

Although the belt drive CVT remains to be a considerable research interest among the mechanical design community and automotive industries, it falls short of the expectations given to it due to its poor endurance performance. People started looking at its successor: the half-toroidal CVT which has been being researched and improved since its first commercial debut in 1999 [11].
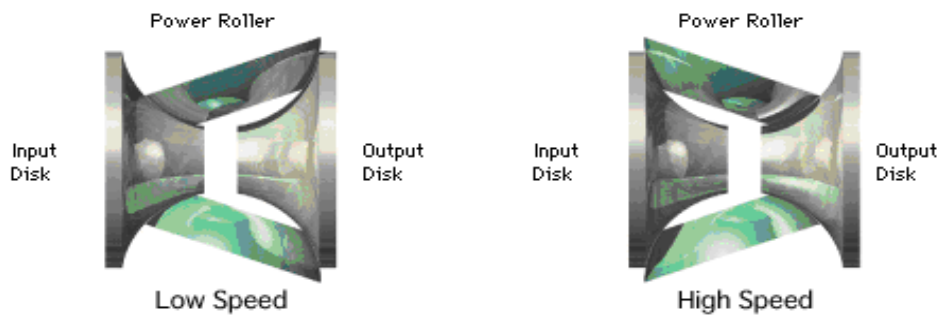
Figure 2: A Half-Toroidal Continuously Variable Transmission

Figure 2 illustrates the components of a half-toroidal CVT. The CVT has a dual-cavity design that combines two sets of input/output disks and power rollers. One disk is connected to the motor while the other to the drive shaft. Two power rollers are mounted symmetrically to the center, inside the cavities, and serve to transmit the power between the input and output disks. Transmission ratio changes are accomplished by tilting the power rollers around the axis of rotation of the trunnions that support the rollers, executing smooth and continuous gear changes. The ratio of the radii of the circles traced by the contact points between the input/output disks and the power rollers corresponds to the tilting of the power rollers and also to the gear ratio. When the radius on the input disk is smaller than output, the input disc rotates faster than the output disc, putting the transmission in low gear. Conversely, when the radius on the output disk is smaller than input, the output disc rotates faster than the input, putting the

transmission in high gear [12].This allows an infinite number of ratio changes within a finite range, raising the power transmission efficiency to the maximum.

Since a half-toroidal CVT has double the number of contact points between the disks and power rollers that transmit force, it is capable of handling higher levels of torque than a conventional CVT. It is different from a general friction drive unit in that it is the power transmission through the intermediary of the oil film and there is no direct contact among the rolling components. It is possible to increase the contact pressure to the limit the rolling component materials. This is why the transmissible torque from a half-toroidal CVT is larger than that of a conventional CVT.

2.3 Adaptive Cruise Control

According to a report from U.S. Department of Transportation, there was an estimate of 6,296,000 police-reported motor vehicle crashes, both fatal and non-fatal, in the United States in 2015 [13]. In those crashes, about 2.44 million people were injured while 35,092 people were killed. A survey conducted by Department of Transportation from 2005 to 2007 pointed out that 94% of the vehicle crashes was related to the drivers while the remaining associated with the vehicles and environments [14]. The specific reasons include driver's inattention, internal and external distractions, inadequate surveillance, speeding, misjudgment of distance, and sleeping. Although significantly improved compared to 2005, the number of vehicle crashes is starting to alarmingly rise back up over the last 5 years.

2.3.1 Conventional Cruise Control

A lot of inventions were researched to help improve the conditions outside and inside the vehicles in traffic. One of those major developments is the conventional cruise control (CCC) which made its debut in automobile industry in 1958 [15]. CCC automatically controls the velocity of the vehicle at a level set by the driver. As CCC reduces the driver's fatigue from keeping track of the vehicle's velocity, it quickly gained attention. The reduction in fuel consumption as a result of utilizing CCC also made it a standard nowadays to have some type of cruise control on a vehicle, as can be seen on a daily basis.
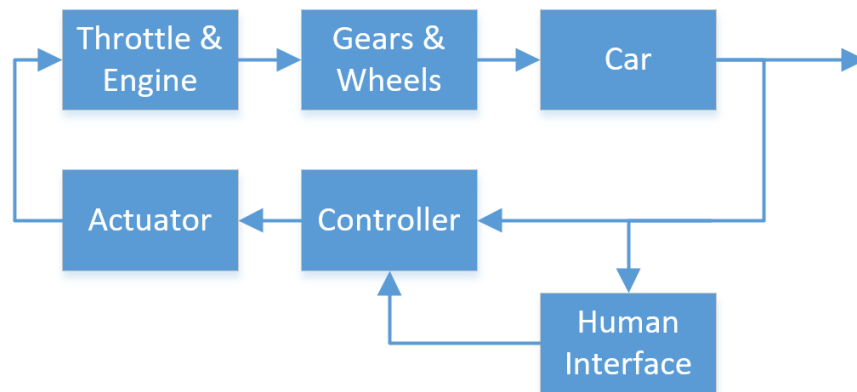
Figure 3: Block diagram of a Cruise Control System

Figure 3 models a typical cruise control system on most cars. Once a desired speed is received from the driver, the controller receives it as input and execute the computations and sends a signal to the actuator unit. The actuator is connected to the throttle valve and controls the throttle butterfly position according to the controller's signal. Under the throttle control, the engine output the torque to the gears, rotating the wheels, and thus moving the car. The current speed of the car is then sent back to the

controller so that it can compute the next signal to send to the actuator. This way, even if the vehicle is moving uphill or downhill, the velocity will still stay unchanged.

2.3.2 Adaptive Cruise Control

One disadvantage of CCC is that it maintains a constant velocity regardless of traffic environments. The answer to that issue is adaptive cruise control (ACC) which is an extension of CCC. ACC first appeared on the Japanese market in 1995 [16]. ACC provides assistance to the drivers in the task of adjusting velocity and maintaining a desired distance from the vehicle in front by controlling the accelerator and the brakes.
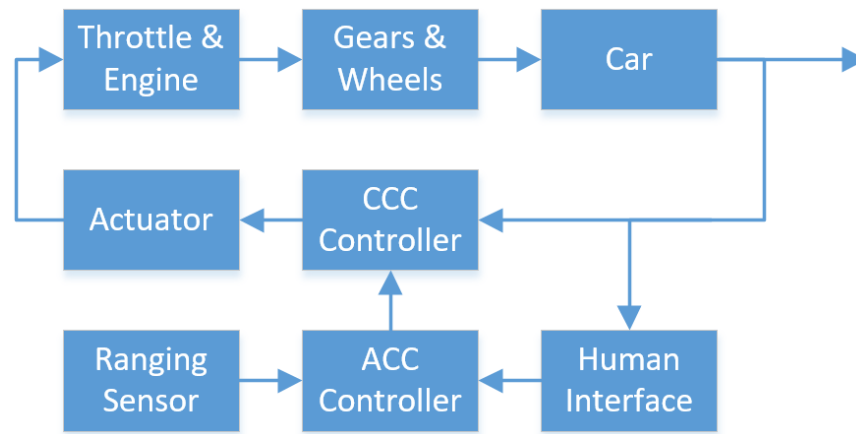
Figure 4: Block diagram of an Adaptive Cruise Control

Figure 4 shows two additional units in the block diagram for CCC. ACC expands CCC on the ability to detect any vehicle ahead and sense the distance between the preceding vehicle and the current car. The information is then transmitted to a controller to process the data based on the implemented ACC algorithm on the system. If no vehicle is detected, the ACC algorithm will output a speed signal to the CCC that matches with the desired speed set by the driver. Otherwise, the ACC algorithm will compute the speed output to keep the minimum safe distance between the car and the vehicle ahead. The CCC controller will receive the signal and execute the procedures of a CCC system.

2.4 Control Lyapunov Function

The problem of solving nonlinear control system remains one of the most challenging subjects in control theory. Nonlinear control problems can be solved by reducing them into Hamilton-Jacobi-Bellman partial differential equations, or Euler-Lagrange open-loop system [17]. But this method is not practical due to the efforts in finding the optimal solutions. Moreover, stabilizing a nonlinear system also poses another difficulty by itself. The theory proposed by Lyapunov successfully alleviated this problem, although there was not a proper procedure in how to apply that theory when it came out. After continuously extensive exploitations of the theory, a design started forming together that can stabilize general nonlinear control systems: control Lyapunov function or [18].

Consider a nonlinear system of the form:

$$\dot{x} = f(x, u)$$

Where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the control vector, and $f: \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}^m$ is locally Lipschitz vector field. A control Lyapunov function (CLF) is a continuously differentiable, proper, positive definite function $V: \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}_+$ such that:

$$\inf_{u \in \mathcal{U}_0} \{\dot{V}(x, u)\} \leq 0$$

Where $\dot{V}(x, u) = V_x \cdot f(x, u) \; \forall x \neq 0$. If it is possible to make the derivative at every point by an appropriate choice of u, then the objective of finding a sufficient and necessary condition for the existence of a smooth CLF is completed and the system can be stabilized with a Lyapunov function under those control actions [19]. That means that if in each state the energy V can be reduced, then it is possible for the energy to be eventually reduced to zero, i.e., it is possible to bring the system to its equilibrium.

2.5 Control Barrier Function

As autonomous systems in various forms start appearing in the market more often, the concerns for safety also rise up. There is no denying that these robotic systems should be restrained by some type of control scheme, which layouts the rules and limits of all possible behaviors of these systems during performance. In that regard, control barrier functions (CBF) are used to guarantee the adherence to the constraints due to their ability to establish invariance of sets, and their relationship to multi-objective control [20].

Consider an autonomous nonlinear system of the form:

$$\dot{x} = f(x)$$

Where $x \in \mathbb{R}^n$ is the state with $x(0) = x_0 \in \mathbb{X}_0$, $u \in \mathbb{R}^m$ is the control input of the system, and $f: \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz vector field. The solutions x of the system are forward complete. A CBF $B(x)$ is a non-negative function with a small value for states far from a constraint and which grows to infinity as the state x approaches the boundary of a set that violates the constraint [21].

To find a suitable CBF, suppose that $h(x)$ is a smooth function which encodes x where $h: \mathbb{R}^n \to \mathbb{R}$. $h(x) > 0$ means the state satisfies the constraints whereas $h(x) < 0$ means the state violates them. The set of admissible states are defined as:

$$\mathcal{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\}$$
$$\partial\mathcal{C} = \{x \in \mathbb{R}^n : h(x) = 0\}$$
$$\text{Int}(\mathcal{C}) = \{x \in \mathbb{R}^n : h(x) > 0\}$$

Where $\mathcal{C} \subset \mathbb{R}^n$, and $\partial\mathcal{C}$ is the boundary of the set $\mathcal{C}$. A locally Lipschitz function $B(x): \mathcal{C} \to \mathbb{R}$ is a CBF if its Lie derivatives $\mathcal{L}_f B(x)$ and $\mathcal{L}_G B(x)$ are locally Lipschitz and if there exist class κ functions α_1, α_2 and $\gamma > 0$ such that $\forall x \in \text{Int}(\mathcal{C})$:

$$\frac{1}{\alpha_1\left(||x||_{\partial\mathcal{C}}\right)} \leq B(x) \leq \frac{1}{\alpha_2\left(||x||_{\partial\mathcal{C}}\right)}$$

$$\inf_{u \in \mathcal{U}}\left[\mathcal{L}_f B(x) + \mathcal{L}_G B(x)u - \frac{\gamma}{B(x)}\right] \leq 0$$

By enforcing this condition, forward invariance of C can be guaranteed and constraint violation can be prevented.

3 Project Goals

3.1 Basic Goals

The following criteria were set for measuring the basic completion of this project:

- The car can drive a straight line without speeding,
- The car can follow a vehicle ahead without crashing,
- The car can smoothly accelerate and decelerate,
- The car can stay in-lane at a low speed (v<15 km/h),
- Design and build the CVT.

3.2 Desired Goals

The following criteria were set for measuring the desired success of this project:

- The car can drive a straight line at medium speed (15 km/h v<30 km/h),
- The car can autonomously shift between low and high gears,
- The car can switch lane safely and smoothly,
- The car can drive smoothly at low speed with CVT.

3.3 Stretched Goals

The following criteria were set for measuring the overachievements of this project:

- The car can drive a straight line at high speed (x30 km/h),
- The car can switch between aggressive and cautious behaviors on traffic,
- The car can recognize lanes in different lightings, weathers, and traffic conditions,
- The car can use anti-lock braking system,
- The car can compute the next state of behavior within 0.1 seconds,
- Control and maintain the car's speed condition with CVT.

4 Design and Analysis

4.1 LIDAR Selection

The car was originally built after Penn's racer car project by a group of PhD students as described above. However, the old LIDAR was broken when we started working on this project. We then faced this problem of whether we should purchase the same model or a different one.

We did some researches to find out the most common LIDAR sensors for academic uses, and then decided to choose Hokuyo UST-10X. Below (Figure 5) is the chart that compares three different LIDAR we have considered.
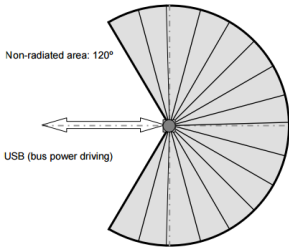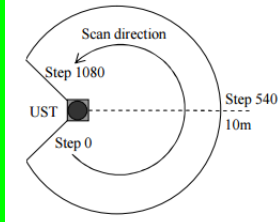
Type	RPLIDAR A2 360° Laser Scanner (The cheapest fully-functional LIDAR existed)	Hokuyo 04LX-UG01 LIDAR (Previously owned one)	Hokuyo UST-10LX LIDAR (Used by MIT's RC Car)
Figure			
Distance Range (m)	0.15 – 6	0.2 – 5.6	0.06 – 10
Angular Range (Degree)	360	240	270
Angular Resolution (Degree / count)	0.45 - 1.35 (Normal: 0.9)	0.36	0.25
Accuracy	< 1.5m: 0.5 mm >1.5m: 1% of the distance	<1m: $\pm$30 mm; >= 1m: 3% of distance	$\pm$40 mm
Scan Rate (Hz)	5 - 15	40	40
Scan method	Evenly mapped	Evenly mapped	Unevenly mapped
Scan Figure			
Price	$ 500	$ 1,100	$ 1,700
Rate	Not recommend for the project	Good to use	The best choice

Figure 5: Comparison between Different Types of LiDAR

4.2 High-Level Software Designs

4.2.1 Localization

Knowing its location in an area is crucial for an autonomous car. In this project, we implemented two methods to localize the car. In our first implementation, we used Gmapping together with AMCL (Adaptive Monte Carlo Localization) to generate a map of the hallway outside the CIBR lab and tested the localization there. In order to use Gmapping, we need LiDAR data and odometry data. The car does not have encoder attached to steering or driving motor. We installed two RPM sensors instead to record the shaft rotations. The sensor was connected to the end of the shaft. Since one full revolution of the shaft takes three full revolutions of the wheels, the accuracy was not good enough and it could not measure the steering angle. We used the data to calculate the translational speed of the car and feed it back to low-level controller. We then decided to use odometry provided by Zed camera which is calculated based on depth image.

4.2.1.1 Transformation

We used ROS Gmapping package to create the map. There are three things the package requires to run, LIDAR data, odometry, and transformation (TF). TF is a concept that has been used a lot in ROS. Basically TF is a tool that links all the coordinates together so we can easily calculate the relative position of one part of the robot to another. In this implementation, the odometry was provided by ZED camera, which was installed in front of the LIDAR. We need to provide the transform of coordinates in order to get precise map from Gmapping.

There are some commonly used frames such as world frame, map frame, etc. There is no clear definition of what exactly these frames mean in some of the packages we used. We came up without own interpretations after some trials. Map frame is the frame that stays fixed. It can be considered as the frame of origin. Odometry frame is attached to the car. It forms a dynamic transform from the map frame.

The ZED software would start publishing a TF when it is launched. We then swapped a few transform from map frame and added a transformation from ZED to LIDAR.

The second method was to use Vicon motion capture system. Motion capture systems are widely used in movie industry, medical and engineering fields. The Vicon system uses multiple optical cameras to track the motions of markers to calculate their positions. The Vicon system is easy to use and provide very precise results.

4.2.2 Adaptive cruise control

The first thing we need to consider in autonomous driving car is the safety. The car needs to drive safely within speed limit to avoid speeding and adapt front car's speed to avoid crash. In this case we implemented adaptive cruise control theory. It is a method that combines the CBF and CLF through quadratic program(QP) [22]. CLF provides the soft constraint to make the car trying to reach the desire speed, and CBF provides the hard constraint to keep a safe distance to the front car and a comfortable acceleration and deceleration. In this case, we assume the road is straight and the car either driving forward or stop. The sensor needed for implementing this method is a 2D LiDAR with a RPM sensor.

4.2.2.1 LIDAR Filter
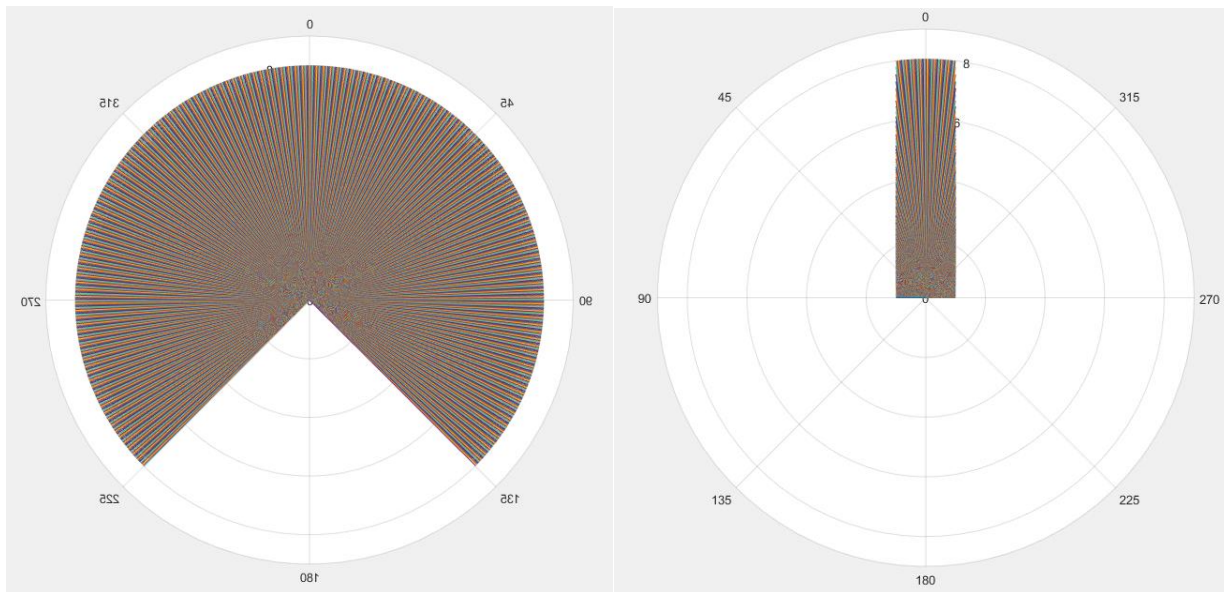
Figure 6: Left – LiDAR Full Range. Right – LiDAR Filtered Range

Since ACC does not control steering, most of the data provided by our LiDAR with 270° range would be not be useful. For example, we would not want the car to stop if there's an obstacle on its side. We also would not want to keep data of objects that are too far away. The main function of the LiDAR is to sense the distance between itself and the front vehicle. Due to this reason, we wrote a Python script to filter out all the unnecessary data. As can be seen in Figure 6, we only keep the data within a 40 cm by 8 m rectangle in front of the origin. From the filtered range of data, we then select the minimum number to make sure the car would not crash into anything.

4.2.2.2 ACC Controller

This method we used was first introduced in the article [22]. The goal of CLF and CBF here seem contradicting. CLF tried to make the car run faster, but CBF would prevent the car from speeding up under certain circumstances. The author solved this problem by introducing the softness constant sigma. This allowed the CLF to be more flexible so that it could work together with CBF.

The controller we used was as following [22].

$$u^*(x, z) = \quad \text{argmin} \quad \frac{1}{2}\mathbf{u}^T H_{acc}\mathbf{u} + F_{acc}^T\mathbf{u}$$

$$\mathbf{u} = \begin{bmatrix} u \\ \delta_{sc} \end{bmatrix} \in \mathbb{R}^2$$

$$\text{(ACC QP)}$$

$$\text{s.t.} \quad A_{clf}\mathbf{u} \leq b_{clf} \qquad \text{(CLF)}$$

$$A_{cbf}\mathbf{u} \leq b_{cbf} \qquad \text{(CBF)}$$

$$A_{cc}\mathbf{u} \leq b_{cc} \quad \text{(CC)}$$

Since the car only drives in a straight line, we can consider a simple model:

$$m \cdot \frac{dv}{dt} = F_w - F_r,$$

Where m is the mass of the car, F_w is the wheel force and Fr is the resistance force. The CLF we used here is:

$$V = y^2,$$

Where

$$y = v_{car} - v_{desired},$$

and the derivative of V is:

$$\dot{V} = 2y\dot{y}$$

Let $x = (P_{car}, V_{car})$ where P_{car} is the cars position and V_{car} is the car's velocity. We can get the dynamics of the system:

$$\dot{x} = \begin{bmatrix} V_{car} \\ -\frac{1}{m} \cdot F_r \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \cdot u(x, z)$$

$$\dot{z} = V_{front} - V_{car}$$

And the A_{clf} and b_{clf} as:

$$A_{clf} = \begin{bmatrix} -\frac{2y}{m} F_r(y) + c \cdot y^2, -1 \end{bmatrix},$$

$$b_{clf} = [\frac{2y}{m}]$$

For the CBF, we can represent the function $B(x)$ as:

$$B(x) = -\log\left(\frac{h(x)}{1 + h(x)}\right),$$

$$where\ h(x) = 1.8x, and\ x = V_{car}$$

Here $h(x)$ is the function describing minimum safety distance, and it is proportional with the car's velocity.

Then we can get:

$$\dot{B}(x, z, u) = f_b(x) + g_b(x) \cdot u,$$

and

$$f_b(x) = -\frac{1.8F_r(x) + m(V_{front} - V_{car})}{m(1 - 1.8V_{car} + z)(-1.8V_{car} + z)},$$

$$g_b(x) = \frac{1.8}{m(1 - 1.8V_{car} + z)(-1.8V_{car} + z)}.$$

Thus we can get A_{cbf} and b_{cbf} as:

$$A_{cbf} = [g_b(x) \quad 0],$$

$$b_{cbf} = f_b(x) + \frac{\gamma}{B(x,z)}$$

For the force constrain:

$$A_{cc} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}$$

$$b_{cc} = \begin{bmatrix} c_a mg \\ c_d mg \end{bmatrix}$$

Where c_a and c_d are the constants for acceleration and deceleration.

We used quadratic program to calculate the output u as force:

$$A = \begin{bmatrix} A_{clf} \\ A_{cbf} \\ A_{cc} \end{bmatrix}, B = \begin{bmatrix} b_{clf} \\ b_{cbf} \\ b_{cc} \end{bmatrix}, H_{acc} = 2\begin{bmatrix} \frac{1}{m^2} & 0 \\ 0 & p_{sc} \end{bmatrix}, F_{acc} = -2\begin{bmatrix} \frac{F_r}{m^2} \\ 0 \end{bmatrix},$$

We can get the result as:

$$v_{next} = v + \left(\frac{u_0}{m}\right) * \delta t$$

where m is the mass of the car

The resistance model of cars is usually determined with tests and measurements. In the paper it was determined empirically. Our car, however, had very different resistance model from common cars.

Testing the resistance model of the car was beyond the scope of this project. Eventually we tried a few reasonable models and stick with the one that suit the system the most.

4.2.3 Trajectory Generation & Trajectory Tracking Controller

With the car able to drive with guaranteed safety, we also need algorithms that make the car follow a non-straight track. We divide the overall architecture into 4 layers: waypoint generator, trajectory generator, trajectory tracking controller, and low-level controller and actuator for the car. The waypoints generator provides a list of waypoints that the car plans to pass and its location as the starting point. The trajectory generator takes the list of waypoints and generate a smooth and feasible trajectory that allow the car to follow. The trajectory tracking controller takes one state from the trajectory and calculate the steering angle and speed for the car to achieve, and low level controller and actuator take the angle and speed and send the command to motor and servo. In this project, we focused on the trajectory generator and trajectory tracking controller.

4.2.3.1 Trajectory generation

In order to generate a smooth and concise trajectory, we used a trajectory generator called minimum-jerk trajectory generator. Jerk is the derivative of acceleration of the desired trajectory, affecting the control inputs and stabilization of the whole system [23]:

$$\vec{j}(t) = \frac{d\vec{a}(t)}{dt} = \dot{\vec{a}}(t) = \frac{d^2\vec{v}(t)}{dt^2} = \ddot{\vec{v}}(t) = \frac{d^3\vec{r}(t)}{dt^3} = \dddot{\vec{r}}(t)$$

Where $\vec{a}$ is acceleration, $\vec{v}$ is velocity, $\vec{r}$ is position, and t is time.

The reason we use minimum jerk trajectory generator is we want to minimize the changing of acceleration along the reference trajectory. Thus, the car will not experience sudden spike in acceleration and deceleration.

We use cubic interpolation splines to connect waypoints to generate a time-based trajectory with each desired state X_d contains $(x, y, \theta, v, \omega)$, where x and y are the desired position, θ is the angle that the car should point to, v is the desired linear velocity that the car should reach and θ is the angular velocity that car should reach. We created two cubic polynomials for x and y that were based on time:

$$a_0 t^0 + a_1 t^1 + a_2 t^2 + a_3 t^3 = f_x(t)$$
$$b_0 t^0 + b_1 t^1 + b_2 t^2 + b_3 t^3 = f_y(t)$$

Where t is time, a_0, a_1, a_2, a_3 and b_0, b_1, b_2, b_3 are constants that represent the trajectory.

We use the speed to constrain the derivative of $f_x(t)$ and $f_y(t)$:

$$\partial x_0 \cdot \sin(\theta) - \partial y \cdot \cos(\theta) = 0$$
$$\partial x_0 \cdot \cos(\theta) + \partial \cdot \sin(\theta) = v$$

4.2.3.2 Trajectory Tracking Controller

In order to follow the trajectory smoothly and safely, we need a controller to control both the speed and the steering angle of the car. After our research, we found a paper that uses control Lyapunov function to build a trajectory tracking controller for UAV (unmanned aerial vehicle). The model we used for our controller is Dubins' car model shown as Figure 7:
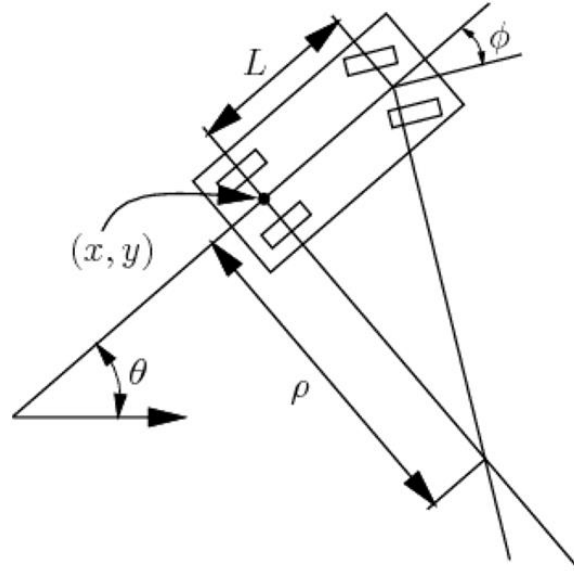
Figure 7: Dubins' Car Model

We can represent this model in equation shown as below:

$$\partial x = v \cdot \cos(\theta)$$

$$\partial y = v \cdot \sin(\theta)$$

$$\partial \theta = \frac{l}{v} \cdot \tan(\Phi)$$

Where x and y are the car's global coordinates, θ is the angle that car currently pointing to, v is the car's current velocity, Φ is the car's steering angle, and l is the length between the front wheel and back wheel.

Then transform the tracking errors expressed in the inertial frame to the car's frame, the error coordinates can be demoted as [24]:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}$$

We followed the paper and get the result for the changing rate of error as:

$$\dot{\bar{x}}_0 = \frac{\omega_r x_2}{\pi_1} + \frac{v_r \sin(x_0)}{\pi_1} - \frac{x_1^2}{\pi_1^3} v_r \sin(x_0) + mu_0 - \frac{x_2}{\pi_1} u_0 - \frac{x_1 x_2}{\pi_1^3} u_1$$

$$\dot{x}_1 = (\omega_r - u_0)x_2 + v_r \sin(x_0)$$

$$\dot{x}_2 = -(\omega_r - u_0)x_1 + u_1$$

Where

$$(\dot{\bar{x}}_0, x_1, x_2) \triangleq (\theta_e, y_e, -x_e)$$

$$u_0 \triangleq \omega_r - \omega^c, u_1 \triangleq v^c - v_r \cos(x_0)$$

$$x_0 = \frac{\bar{x}_0}{m} - \frac{x_1}{m\pi_1}, \pi_1 \triangleq \sqrt{x_1^2 + x_2^2 + 1}$$

Then we can get the f and g function for our dynamic system as:

$$\dot{x} = f_1(t, x) + g_1(t, x)[u_0, u_1]^T$$

Where

$$x = [\bar{x}_0, x_1, x_2]^T,$$

$$f_1(t, x) = \begin{bmatrix} \frac{x_2}{\pi_1} \omega_r + \frac{1 + x_2^2}{\pi_1^3} v_r \sin\left(\frac{\bar{x}_0}{m} - \frac{x_1}{m\pi_1}\right) \\ x_2 \omega_r + v_r \sin\left(\frac{\bar{x}_0}{m} - \frac{x_1}{m\pi_1}\right) \\ -\omega_r x_1 \end{bmatrix}$$

$$g_1(t, x) = \begin{bmatrix} m - \frac{x_2}{\pi_1} & -\frac{x_1 x_2}{\pi_1^3} \\ -x_2 & 0 \\ x_1 & 1 \end{bmatrix}$$

We get the result $u = [u_0, u_1]^T$ as:

$$u_0 = \text{sat}\left(-\eta_\omega \bar{x}_0, \underline{\omega}_{min}, \overline{\omega}_{max}\right)$$

$$u_1 = \text{sat}\left(-\eta_v x_2, \underline{v}_{min}, \overline{v}_{max}\right)$$

Where u_0 is angular acceleration, and u_1 is linear acceleration. Here η is a constant that both satisfies control Lyapunov function and saturation inputs [24].

And the command output is:

$$\Phi = \tan^{-1}\left(\frac{(\omega_r - u_0)v}{l}\right)$$

$$v = v_c - u_1$$

Where Φ is the steering angle, and v is the car's speed.

4.3 Low-Level Software Designs

4.3.1 Low level motor control

The RC car is originally controlled by a radio controller. It can send driving forward, reverse, left turn and right turn by encoding those commands to radio signal, then emit it to the RC car. The radio receiver on the car can decode the signal to two different channels of PWM signals, which are speed and steering respectively, to ESC. Finally, the ESC will translate the PWM signals to certain electrical power to drive the motors. The RC car signal control is shown in Figure 8.
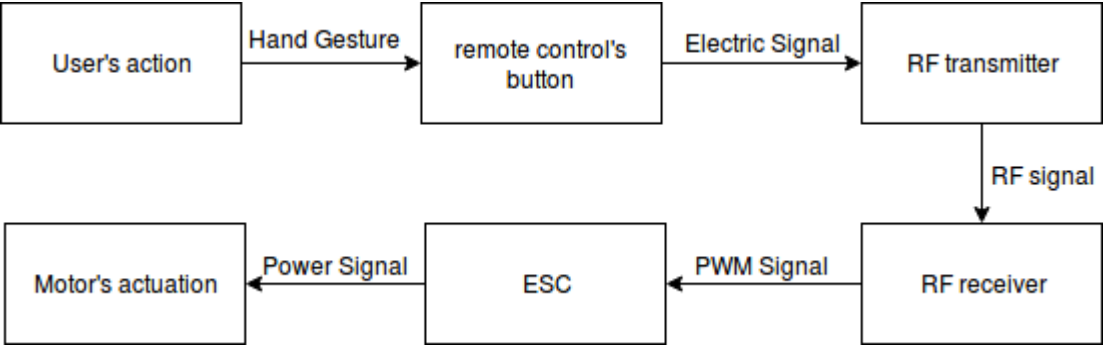
Figure 8: Diagram of RC Car signal control

To control the motor in the customized speed, we need to hack into this process. By doing some research, we planned to emulate the PWM signals and control the ESC.

We analyzed the encoded PWM signals from two channels, and found out that both of them has 100 Hz frequency. The steering angle and speed are controlled by duty cycle: from 10% to 20%. When the duty cycle is 10%, it indicates the max reverse in driver and max left turn in steering, and when the duty cycle is 20%, it indicates the max forward in driver and max right turn in steering. In this range, the percentage of duty cycle is evenly mapped with corresponding driver and steering behaviors. By following this regulation, the corresponding duty cycle percentage of stall mode in driver and straight mode in steering is 15 percent. Based on all findings, we thought that the signal was easy to be duplicated and hacked using a low-cost microprocessor.
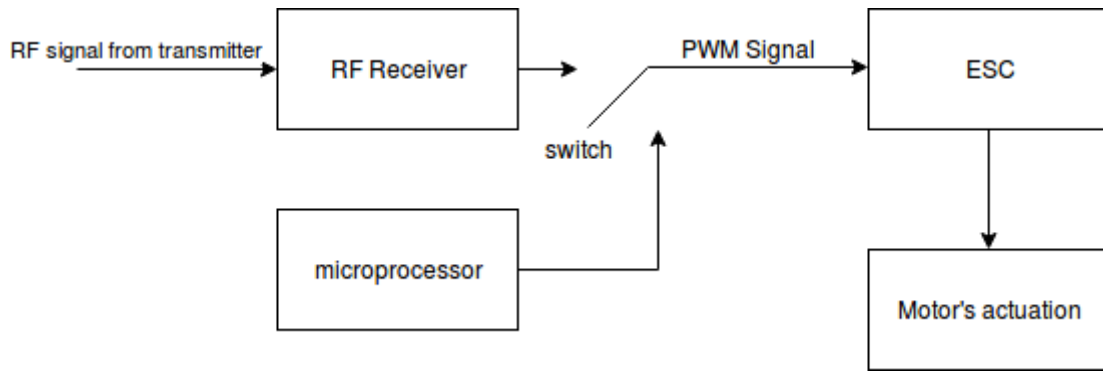
Figure 9: Diagram of RC Car signal control

To enable the RC car that was controlled by either RC controller or the microprocessor, we embedded two switches so the signal sources can be easily switched. The final diagram signal control is shown in Figure 9.

4.3.2 Choice of processor

Having an NVIDIA TX2 Board, we decided to use it as the main processing unit. Although TX2 has capability to output the required PWM signal, we also equipped a Teensy 3.2 Board as an additional lower-level motor controller for the following reasons:

- Make the system design clear by separating lower-level control and higher-level control, causing the system easier to be tested and debugged separately.

- Decrease the risk. If the low-cost Teensy was broken or brunt, it wouldn't impact the TX2, which costs much higher than Teensy.

- Unify the protocol. The communication between Teensy and TX2 is through serial, which is supported by ROS serial package. This design saves extra works from transcript messages between different protocols.

- Easy wiring. Since TX2 was mounted on the top layer of the RC car while the ESC was mounted on the second layer, putting the Teensy closer to ESC can make the wiring more elegant.

4.3.3 Lower-level Controller

Since our car is modified with ABS (Acrylonitrile butadiene styrene), LiDAR, Zed camera and TX2, the car is overweight and the original speed control can't control the speed as expected. We used a PD controller to control the speed of the car.

4.4 Mechanical design

4.4.1 Car Base Case and Material Choice

4.4.1.1 Material

Originally this car base case is made of Acrylic, which made this car vulnerable and so easy to be broken. Since this car case is already broken at some part, we decide to change the whole case that made out of ABS, based on the high tensile strength properties of ABS and availability of cutting in Washburn.

Table 1 Table of ABS Material Properties [25]

Mechanical Properties	Metric	English	Comments
Hardness, Rockwell R	68.0 - 113	68.0 - 113	Average value: 102 Grade Count:56
Ball Indentation Hardness	65.0 - 110 MPa	9430 - 16000 psi	Average value: 93.2 MPa Grade Count:11
Tensile Strength, Ultimate	23.0 - 49.0 MPa	3340 - 7110 psi	Average value: 38.2 MPa Grade Count:26
Tensile Strength, Yield	13.0 - 65.0 MPa	1890 - 9430 psi	Average value: 41.3 MPa Grade Count:100
⚠	22.1 - 59.3 MPa @Temperature -18.0 - 71.0 °C	3210 - 8600 psi @Temperature -0.400 - 160 °F	Average value: 40.7 MPa Grade Count:1
Elongation at Break	3.00 - 150 %	3.00 - 150 %	Average value: 33.8 % Grade Count:67
Elongation at Yield	0.620 - 30.0 %	0.620 - 30.0 %	Average value: 5.69 % Grade Count:47
Modulus of Elasticity	1.00 - 2.65 GPa	145 - 384 ksi	Average value: 2.10 GPa Grade Count:45
⚠	1.50 - 2.60 GPa @Temperature -18.0 - 71.0 °C	218 - 377 ksi @Temperature -0.400 - 160 °F	Average value: 2.05 GPa Grade Count:1

In Table 1, tensile strength of ABS is 23.0-49.0 MPa and modulus of elasticity is 2.10GPA average, which means this material is hard and high elasticity, in other words hard to break.

4.4.1.2 Case design and CAD model

We use SolidWorks to design the structure of Car case during the summer shown in Figure 10. Depend on the previous design of case, we came up with an ABS board that can stand the enough pressure and able to put every device that we need on it.

Based on what we can get from market, we choose 0.28 thickness ABS board for our car case, which is totally enough for the elasticity and hardness. We design our case structure to put ZED camera, lidar and TX2 on. We choose this shape because of beauty and capability to put stuff and give enough space for chassis to put more staff. All the holes on the board is drilled hole for screw. All of manufacture are in Washburn with the Help of Loiselle, James T, who is the Sr Instruct Lab Technician in WPI.
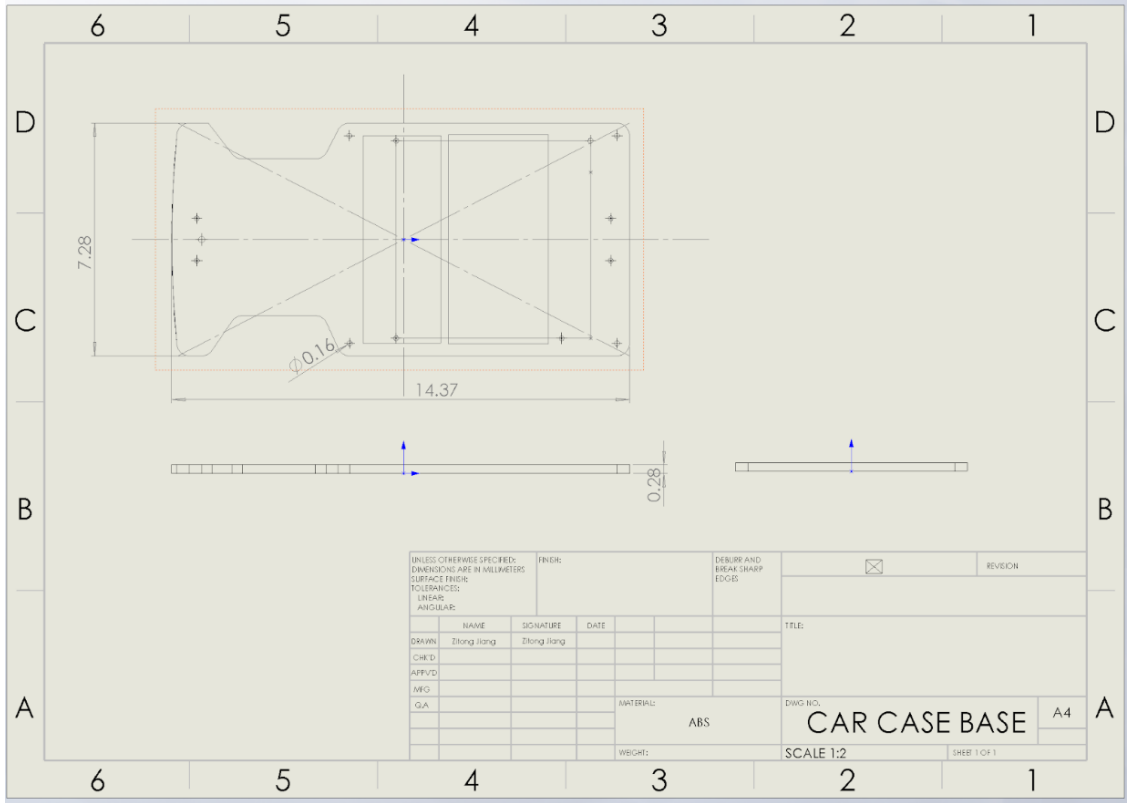
Figure 10: Car Case Base Design Drawing

4.4.2 Transmission Design and Analysis

4.4.2.1   CVT Original design inspiration and specs.

Half toroidal continuously variable Transmission contain Four major parts, Input Disc, output disc, power roller and trunnion. Our original design is based on those aspects and generated them in Solidworks. While the half-toroidal CVT has proven to be superior to traditional transmission system, for this type of CVT to display its unique advantages to their fullest extent, however, the design has to be made with the specifications that provide high efficiency while still satisfying the requirements for vehicle mount ability and durability. There have been many researches on the parameters of the half-toroidal CVTs and constructing predicting models with these parameters. There have also been many studies and analyses on the efficiency of the toroidal CVTs. But there have only been a few researches conducted on figuring out the most optimal configurations of the half-toroidal CVTs to connect the models and the efficiencies together. The half-toroidal CVT and its control system for this project were built with the geometry given in the paper by Delkhosh et al [26]. The reference papers from this paper were also looked into to further the understanding of the paper's model and the simulation used to construct it.
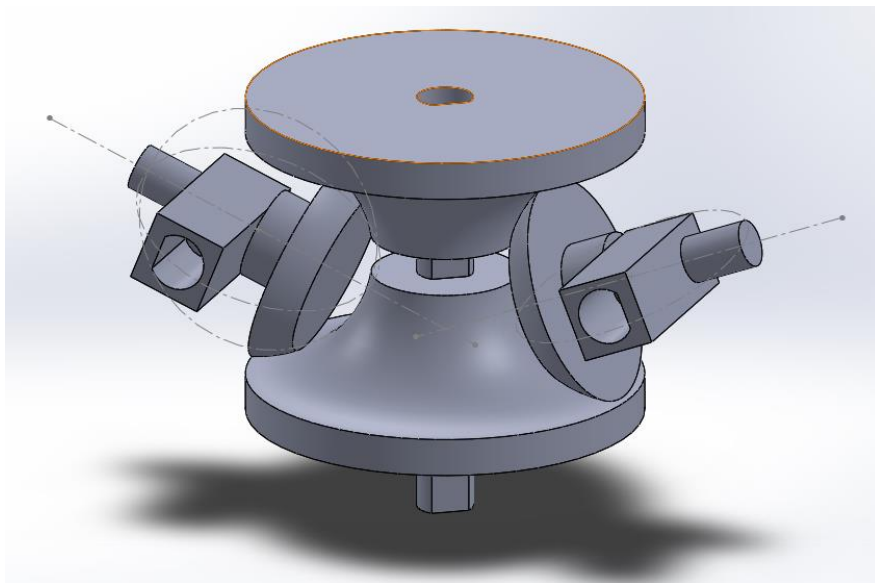
Figure 11: CAD Original Design of CVT

Figure 11 is our original CAD model design. All of those specs and dimensions we gain are from Delkhosh(2011) [26]. This paper maximizes efficiency by optimizing the radius, pressure angle and all the mechanical dimensions. The radius of edge in Power roller has to match input and output radius, which are 0.53 Inches, to have the most touching surface of power roller between input and output to have

enough friction surface. Trunnion matches the dimensions of Power roller as well. The center of Trunnion is exactly the turning center when we assemble them together to best match and reduce vibration.

4.4.2.2   CVT Analysis and Calculations

4.4.2.2.1   Analysis

After our original design of half-toroidal CVT, we discuss with Professor Daniello. We need traction forces acting on the input and output to prevent them from splitting and sliding. Because there will be a large number of forces acting on the roller and disc surfaces due to high RPM input, approximately 6000 RPM (Our input motor, Traxxas 3500R, Brushless motor). Those forces will definitely push the roller and input to other side, in other words they cannot hold in their original position. If we only use outside box as holder to hold two discs, Acrylic box will definitely break.

4.4.2.2.2   Calculation

When we calculate the Traction force, there are two directions, one is normal, and other is Vertical direction. The important one is the vertical one, based on our design and specs of CVT, Vertical one is the forces that push input and output away from each other. In order to calculate the forces, first we need to know the Torque of our motor. Based on the engineering toolbox, we approximately calculate our Torque as $300 \ N \cdot mm$, Rpm we use is 20,000 as input. The reason we use the large number of Rpm is Because we want to have enough traction forces for calculation purposes. We use the equations below to Calculate them. All of our specs are from Figure 12:

$$Torque = \frac{Power \cdot 9.547}{RPM} \ [27]$$

$$Traction \ Force = \frac{Torque}{Radius}$$

$$Tractio \ Force = \mu \cdot Traction \ Load$$

$$Normal \ Direction \ Load = \sin(\theta) \cdot Traction \ Load$$

$$Vertical \ Direction \ Load = \cos(\theta) \cdot Traction \ Load$$
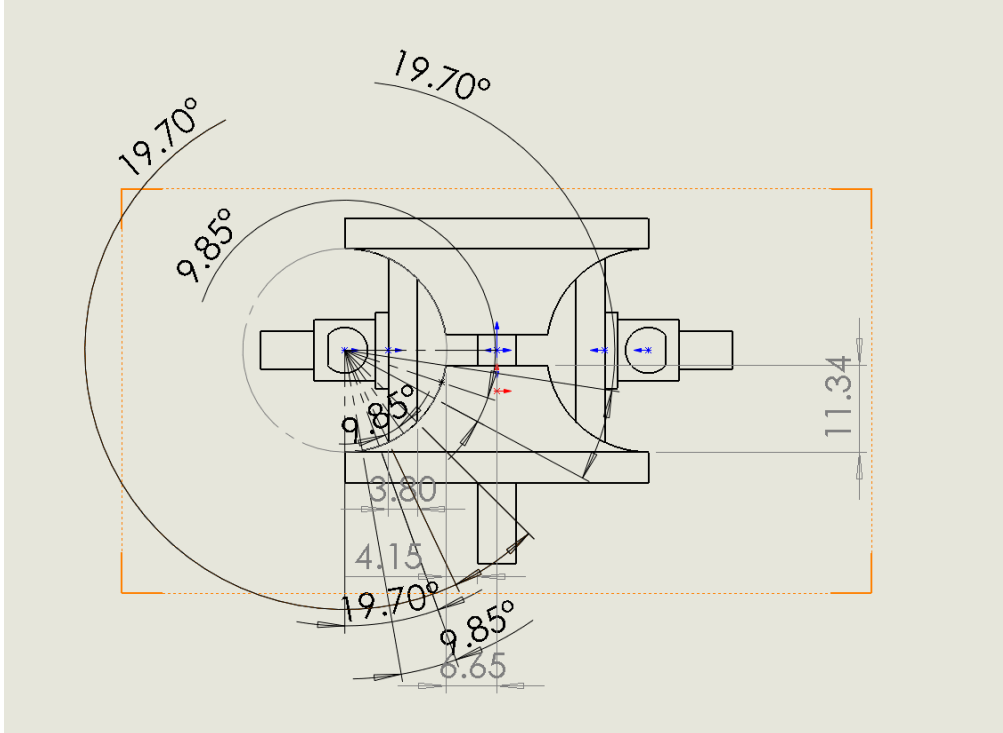
Figure 12: Engineering Drawing of CVT

Table 2 Calculations of Traction Force

Torque($N \cdot mm$)	360		Traction force	Traction load	Normal direction	vertical direction	Torque of step motor	load needed
RPM (approximately)	20000	max	20.5011	20.5011	6.5860	19.4145		1.6465
Power (approximately)	750	max lub	20.5011	68.3371	21.9534	64.7148		5.4883
Rmax	17.56	min	50.0695	50.0695	49.3306	8.5704		12.3326
Rmid	12.16	min lub	50.0695	166.8985	164.4353	28.5682		41.1088
Rmin	7.19	mid	29.6053	29.6053	24.2067	17.0443		6.0517
theta min	18.74	mid lub	29.6053	98.6842	80.6889	56.8144	5.2076	20.1722
theta mid	54.85							
theta max	80.15							
rad min	0.3271							
rad mid	0.9573							
rad max	1.3988							
friction coefficient (no lubrication)	1							
friction coefficient (lubrication)	0.3							
roller diameter	10.91							

Based on those dimensions in Table 2, we calculate our Traction load needed for input and output shown in Table 2, which is in the chart around 6~20 N. In the chart, I assumed our material as Aluminum T6061. Metal to metal touch friction coefficient is 1 [28], from Engineering Toolbox without lubrication is 0.3 as coefficient [28]. Specs of our motor, we use Traxxas website to get them [29].

4.4.2.2.3   Analysis in ANSYS

After our calculation process, we analyze them in ANSYS, we use static structural to analyze the whole part. Because of analyzing such a huge and complicated movement is not very suitable in ANSYS. We use static one instead to see where the maximum stress happens, and how much deformation there

will be. The forces we applied on the input and output are the traction forces that we calculated before, 6 to 20 N, acting on the top surface on input and output.
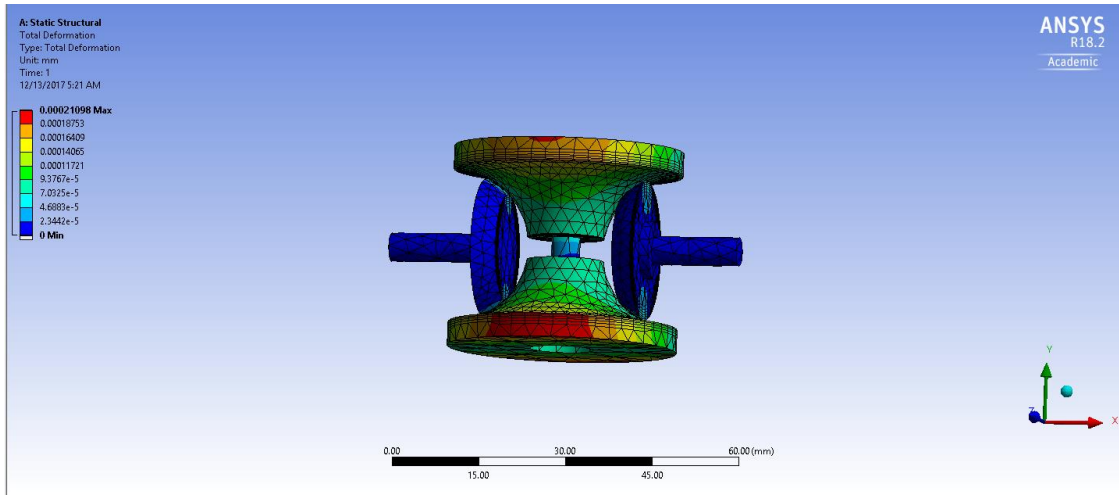
Figure 13: ANSYS Analysis of Deformation of CVT
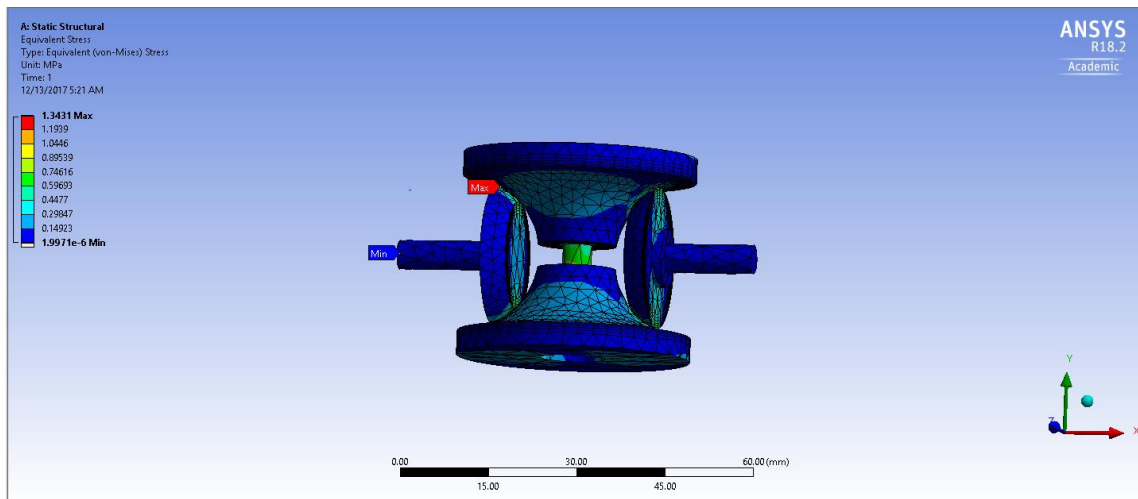
Figure 14: ANSYS Analysis of Equivalent Stress of CVT

As we can see in the ANSYS analysis, deformation is totally not the problem. The Maximum equivalent stress happened at touching point of power roller between input and output. This exactly matches my hypothesis. With the help of ANSYS and my calculation, I finally can improve my design.

4.4.2.3   CAD detailed and Final design.

As I have mentioned before. We need traction forces. Spring is one of the best choice. We found one from MasterCarr. Maximum load and load by inches provide enough load we needed, which is 7.96 lbs., and rate is 23.20 lbs./in. But if we add Spring directly, input and output cannot rotate. We need

another set of thrust bearings to let spring provide load, while input and output can rotate as well. We choose bearings from amazon with good quality and good price. Then in order to let power roller shaft and input and output shaft rotate smoothly, we add ball bearings, from amazon. Between shaft and ball bearings, and bearings and trunnion, we use press fit to hold them. And between shaft of power roller and trunnion, shaft of input and hole on output, I choose loose fit because I need them to have smooth rotation. Most importantly, we need box to hold my design, and for demonstration purposes as well. I choose acrylic as material for box, press fit for ball bearings and box. In order to assemble box together, we choose T-slot and screw and nut. Here is the specs and final version of CVT shown in Table 3 and Figure 15.

Table 3 CVT Specifications

Input and Output Radius	0.53 inches
Power roller Radius	0.53 inches
Box dimension	2.63*2.9*2.15
Gear ratio	0.625:1:1.25
Material	Aluminum
Box Material	Acrylic
Traction force	10~20N
Input motor	Traxxas 3350R
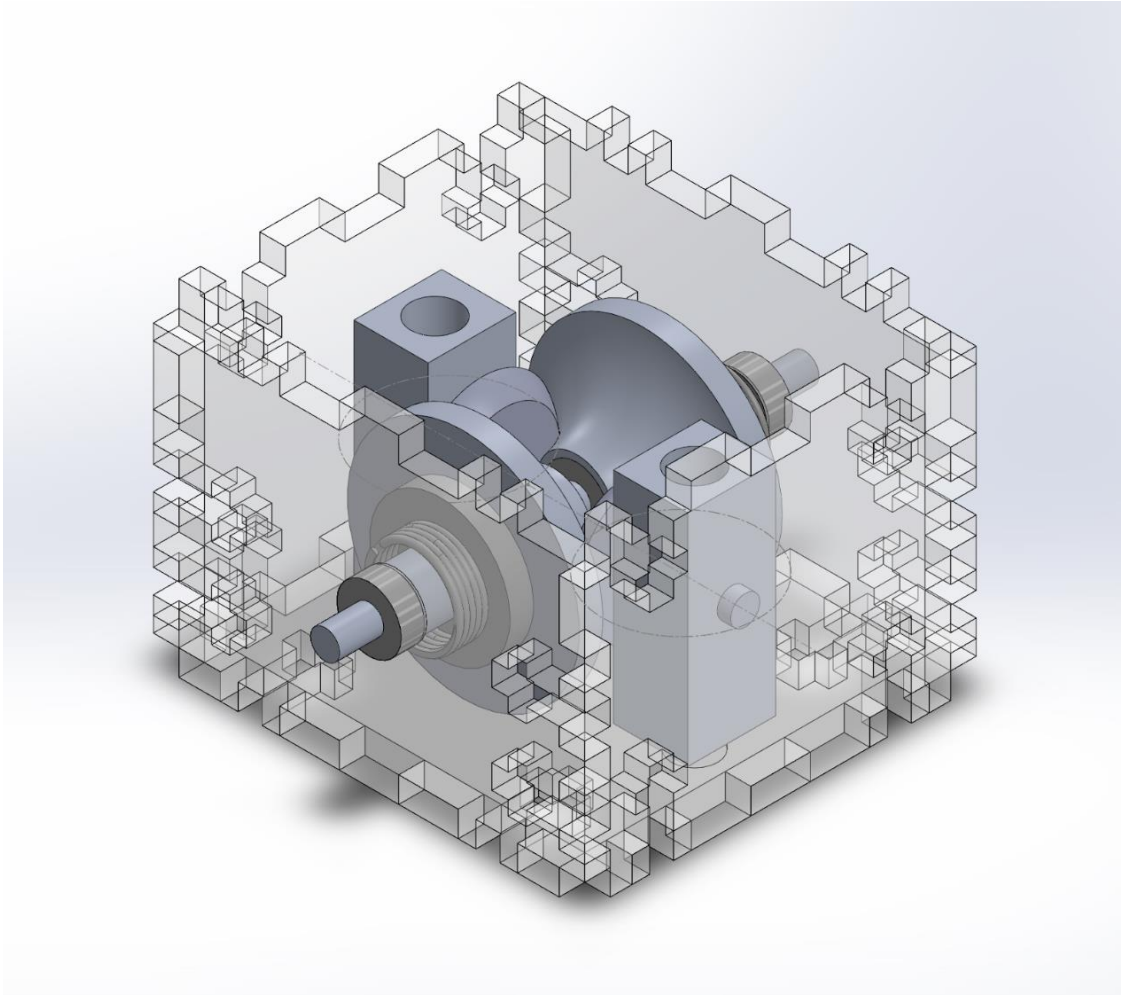
Figure 15: Improved CAD Model of CVT

4.4.3 Manufacture

We manufacture all of our parts in Washburn and Anchor Labs. We gain the material parts directly from Washburn, Special thanks to Ian and James for Great help when Manufacturing. Power roller is manufactured by Lathe in Washburn, trunnion is manufacture by Minimill. For input and output disc, we ask anchor labs to manufacture for us. All the pics of manufacturing details are in Appendix. The press fit we are using RC1 [30], the loose Fit we are using RC3. for set screw we are using No. 8 screw. Here is our final product.
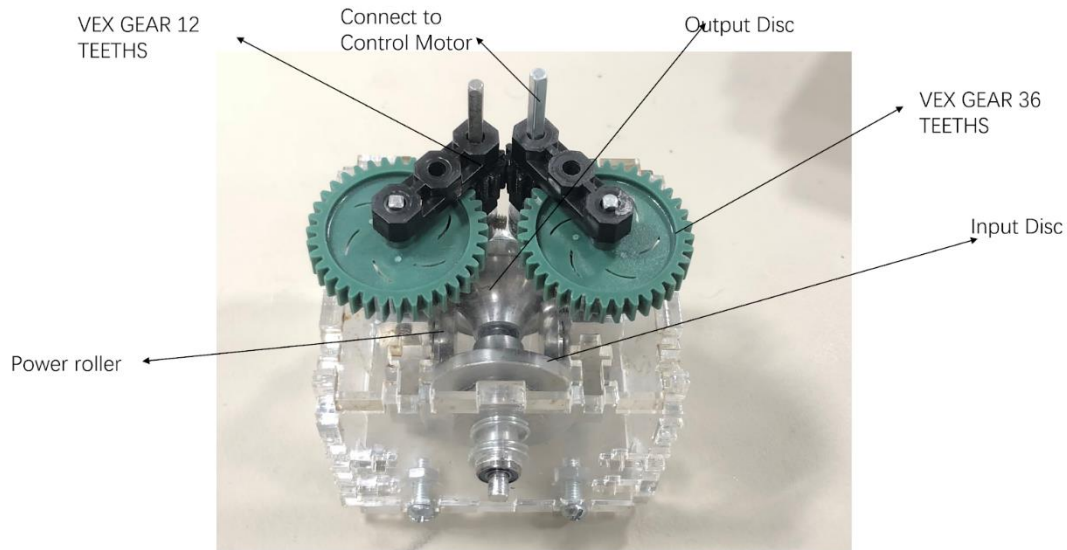
Figure 16: Manufactured Prototype of CVT

In Figure 16, front Disc is the input, back Disc is the output, two rollers one each side are power rollers. On the top are trunnions which have two green vex gears connect to two small black vex gears. Gear ratio on the top is 12:36. The motor that control the angle of power roller between input and output will be connect on one of the two shafts on the top.
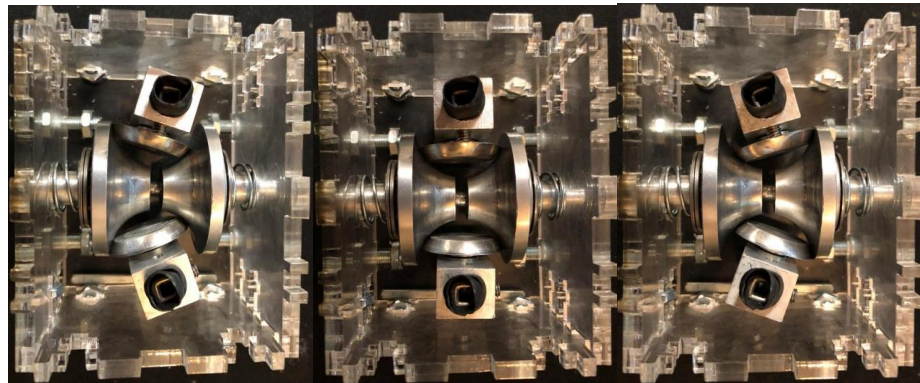
Figure 17 Left: High Gear ratio; Middle: 1v1 Gear ratio; Right: Low Gear ratio

5 Results

5.1 ACC Performance

5.1.1 Simulated Result

In order to test the performance of ACC controller, we simulated the situation with the conditions listed here. We made the desired velocity as 24m/s and made the front car's velocity as 14m/s which is smaller than the desired velocity. We made the car start with the speed of 20m/s and the initial front car distance is set to 100 m. We simulate the situation in MATLAB and get the result shown as plot below.
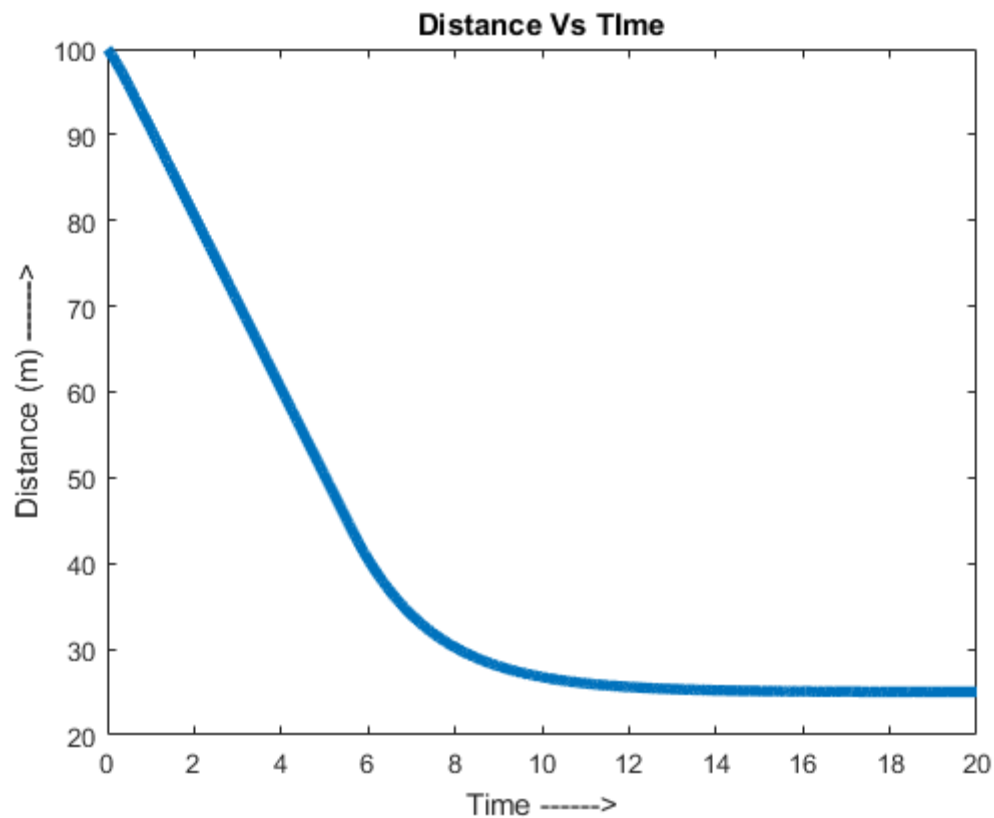
Figure 18: Plot of Simulated Distance Between RC Car and Front Car

As can be seen in Figure 18, the simulation shows that the initial distance between the autonomous car and the car in front is 100m. As the autonomous car approaches, ACC changes the velocity output so that the distance gradually becomes 25m at the end. The slow starts around sixth

second and ends at fourteenth second. ACC system only needs 8 seconds to quickly adapt to the target's velocity.
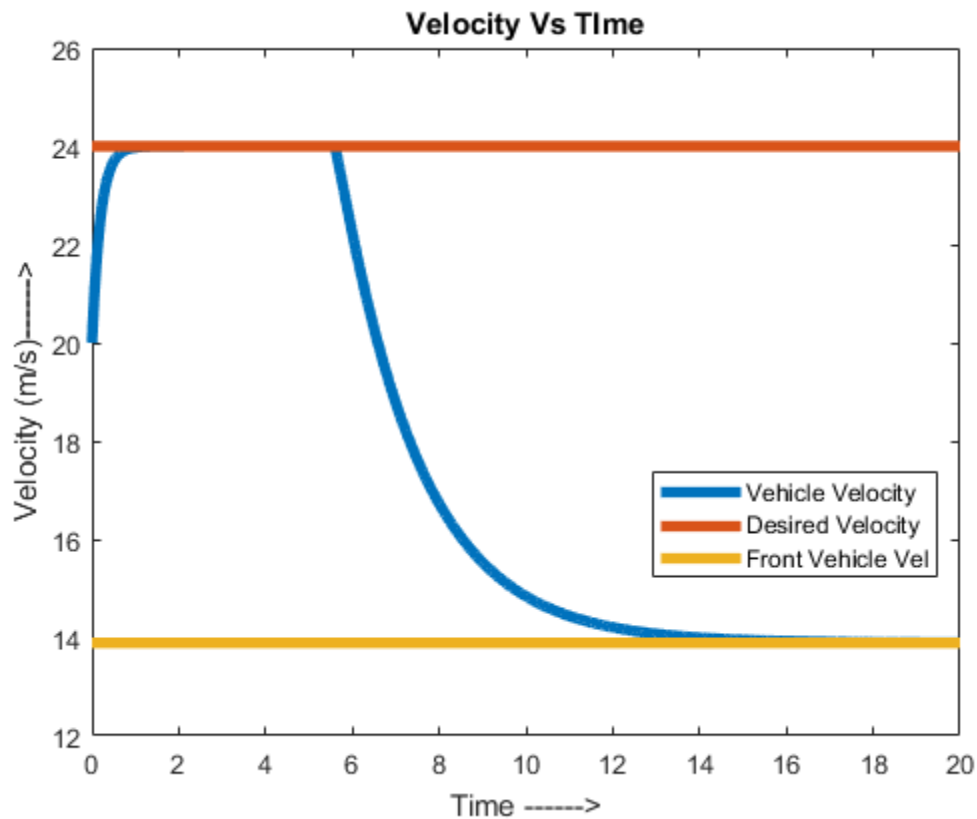
Figure 19: Plot of Simulated Velocities

Figure 19 verifies this notion. The car starts out at 20m/s and reaches the desired speed in around 0.5 seconds. As the velocity stays at the limit from second 2 to second 6, that is when the distance between is still larger than the safe distance of the ACC. When a car comes into range, the RC car starts decelerate quickly at second 6 to match with the front car's velocity at second 14. Given the front car's velocity remain constant, the RC car's velocity stays unchanged as well, and the distance between them stays constant, as seen in Figure 18: Plot of Simulated Distance Between RC Car and Front CarFigure 18.

The simulation result shows that the car is able drive as the desired velocity when it has a very long distance to the front car. The car is able to decelerate and adapt its speed to the front car's speed when it getting close to the front car.

5.1.2 Real World Testing Result

      We tested the car in our 1/10 car. We set the drag force and brake force for the ESC to 90%. We tested the car's performance with a person walking at front with varying speed, and here is the result in Figure 20 and Figure 21..
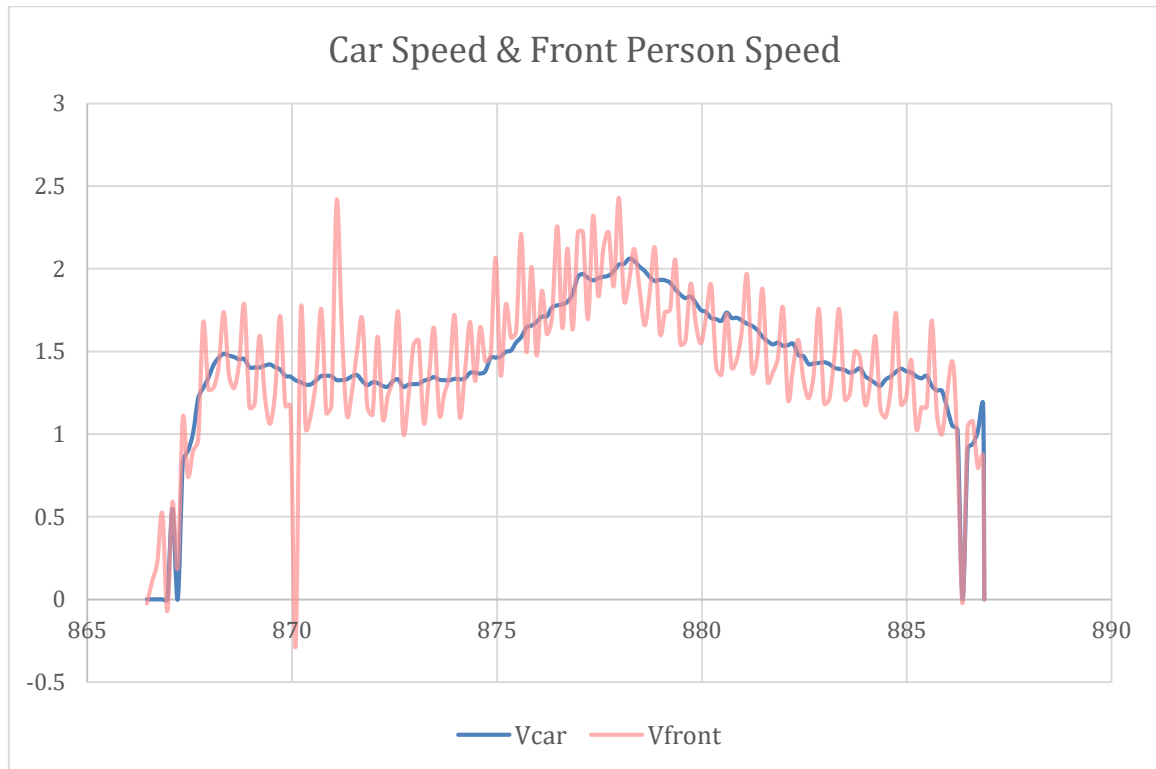
Figure 20: Plot of Real-World Velocities

      Figure 20 shows the velocity of a person walking in front of the car and the velocity of the RC car. On average, the car's velocity stays in-between the peaks of the person's velocity. Because the measurements were taken when the sensor scanned the midpoint between the two legs, the plot has a lot of spikes and noises. Despite that, the adapted velocity was smooth and unaffected even by the drastic drop in velocity of the person.

      From a distance perspective in Figure 21, the RC car keeps a minimum distance with the front car, as can be seen in a constant small space between the blue line and red line. When the person speeds up and walk further away from the car, the car accelerates to the velocity limit until the person falls in the safe range. As the person slows down, the car decelerates to match the person's speed as well. The drastic stop at the end happens because the person already comes too deep inside the safe range. The car then resumes the velocity as the person goes further a bit.
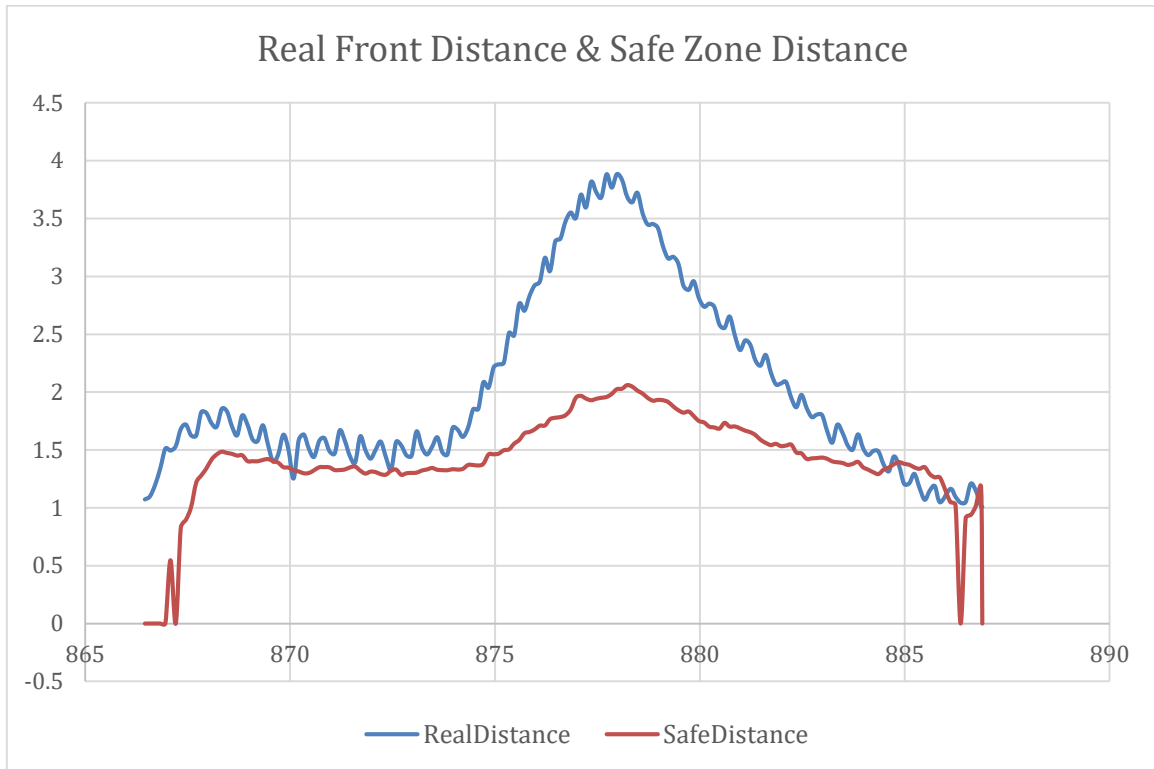
Figure 21: Plot of Real-World Distance Between RC Car and Front Car

The car was able to successfully adapt to speed of the front car without collisions. If speed limit was lower than the front car speed, it would reach the maximum speed and stay there until the front car slows down. If the speed limit was higher than the front car speed, it would keep a safe distance from the front car. From the distance plot, we can see the real distance is always greater or equal to the safety minimum distance.

5.2 Line Tracking Performance

5.2.1 Trajectory Generator

The minimum-jerk trajectory generator was simulated using MATLAB. The result is shown in Figure 22:
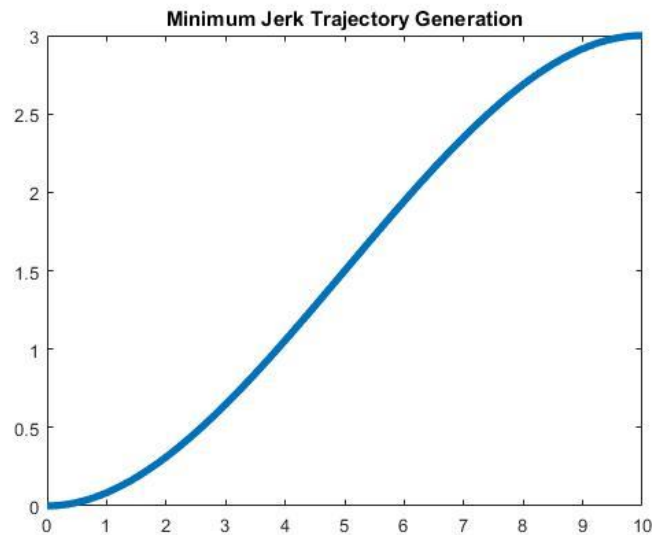
Figure 22: Trajectory Generated by Minimum Jerk Algorithm

The starting point of this trajectory is at (0,0,0) and ending point at (10,3,0) with the format (x,y,z). The trajectory can be represented as shown below:

$$f_x(t) = t$$
$$f_y(t) = (0.09)t^2 + (-0.006)t^3$$

Where $f_x(t)$ is the function that describes the x position while $f_y(t)$ describes the y position. As shown in the graph, the trajectory generated is very smooth and does not cause sudden changes of acceleration.

5.2.2 Trajectory Tracking Controller

5.2.2.1 Simulation Result

We simulated our controller in MATLAB with the referenced trajectory generated by minimum jerk. We calculated the η by using parameters in Table 4:

Table 4 Saturated Factors for Calculating η

Symbol	Description	Value
V_{min}	Minimum Motor Speed	0.0
V_{min}	Maximum Motor Speed	10.0
ω_{max}	Maximum Angular Speed	2.0
δ_v	Maximum Speed Difference Between each state	0.05
δ_ω	Maximum Angular Speed Difference Between each state	0.25

With the values above, we get values η_ω = 2.25 and η_v = 0.775. In order to generate the trajectory that follows a designated path, those values are substituted into the controller:

$$u_0 = \text{sat}\left(-\eta_\omega \bar{x}_0, \underline{\omega}_{min}, \overline{\omega}_{max}\right)$$
$$u_1 = \text{sat}\left(-\eta_v x_2, \underline{v}_{min}, \overline{v}_{max}\right)$$

If the car starts out at the same position as the desired path, the simulated car's trajectory will turn out as followed:
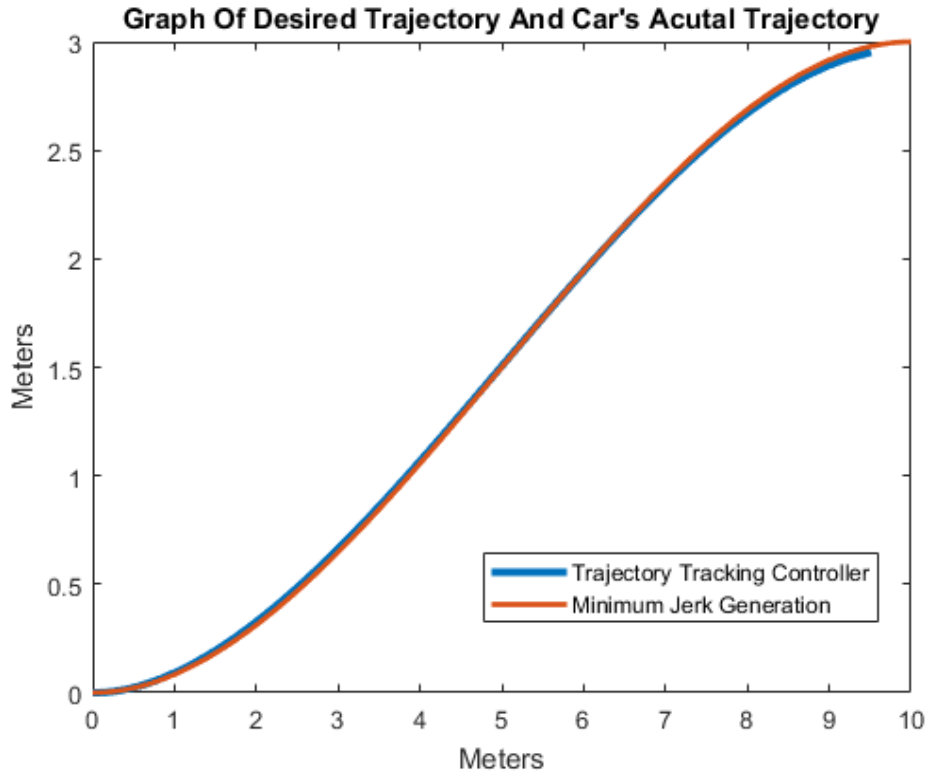
Figure 23: Simulation of Desired Trajectory Compared to Actual Trajectory at the Same Starting Point

As shown in Figure 23, the trajectory generated by the controller follows the desired path very closely and smoothly. The offset between the two trajectories are inconsiderable. Below are the plots of x and y positions of the car compared to the desired positions:
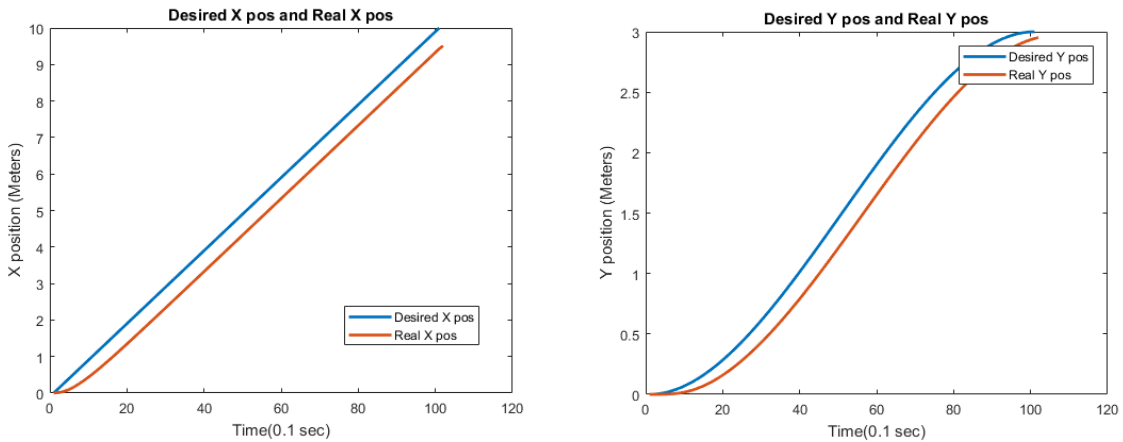
Figure 24: Left – Simulation of Desired x Position Compared to Actual x Position

Right – Simulation of Desired y Position Compared to Actual y Position

In Figure 24, the offsets between the x and y positions are larger than they seem Figure 23. This is because only a portion of the trajectory are shown and zoomed in. Even so, the linear and angular velocities of the car match very well with the expected values, as seen below. There is also a consistent offset between the car's and expected values. This is the errors coming from the estimating nature of the controller which tries to follow a trajectory.
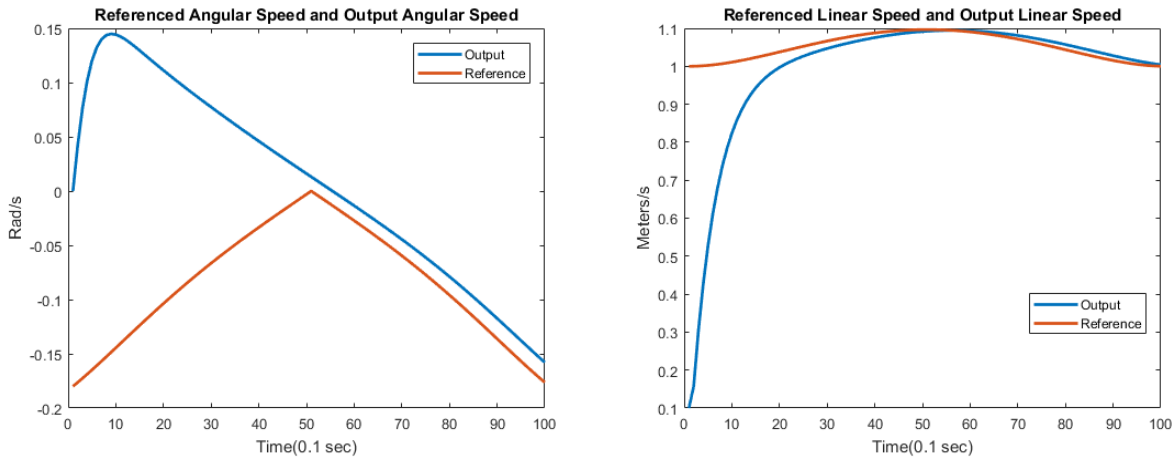
Figure 25: Left – Simulation of Angular Speed

Right – Simulation of Linear Speed

When the starting position of the car and the starting position of the trajectory are the same, the car's trajectory is very close to the referenced trajectory. The output is very smooth and the distance between desired coordinates and car's coordinates are very small and does not have the sign of growing as the time increases.

We also simulated the controller with the car starting at a different position from the trajectory starting point. The result is shown in Figure 26.
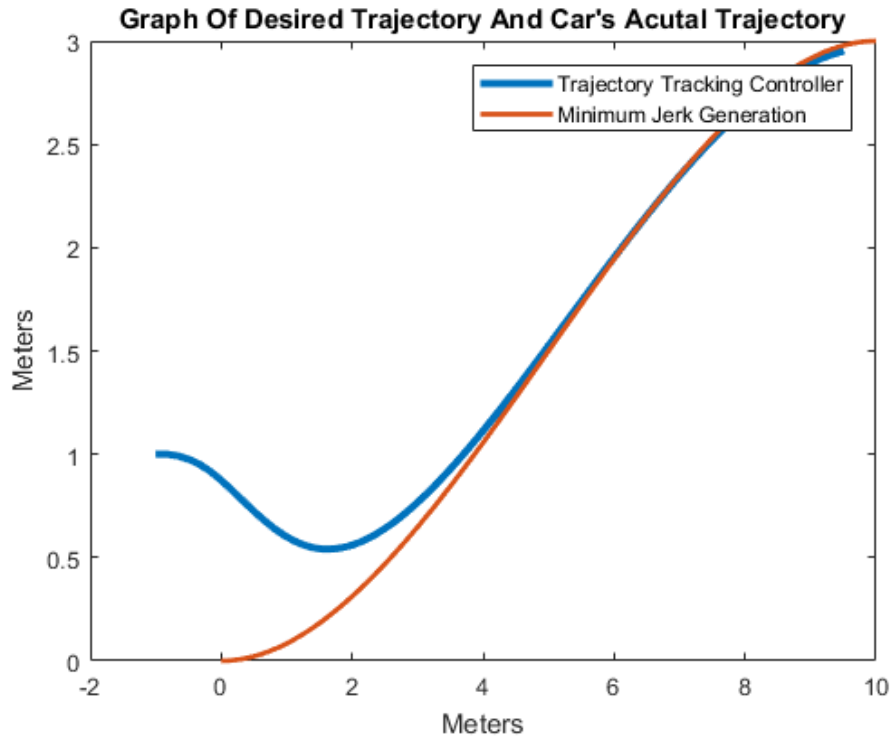
Figure 26: Simulation of Desired Trajectory Compared to Actual Trajectory at Different Starting Point

The trajectory follows the desired path very well. It only takes 2 seconds for the trajectory to converge to the desired path. The offset between the two trajectories after second 4 is also inconsiderable. Below are the plots of x and y positions of the car and the desired positions:
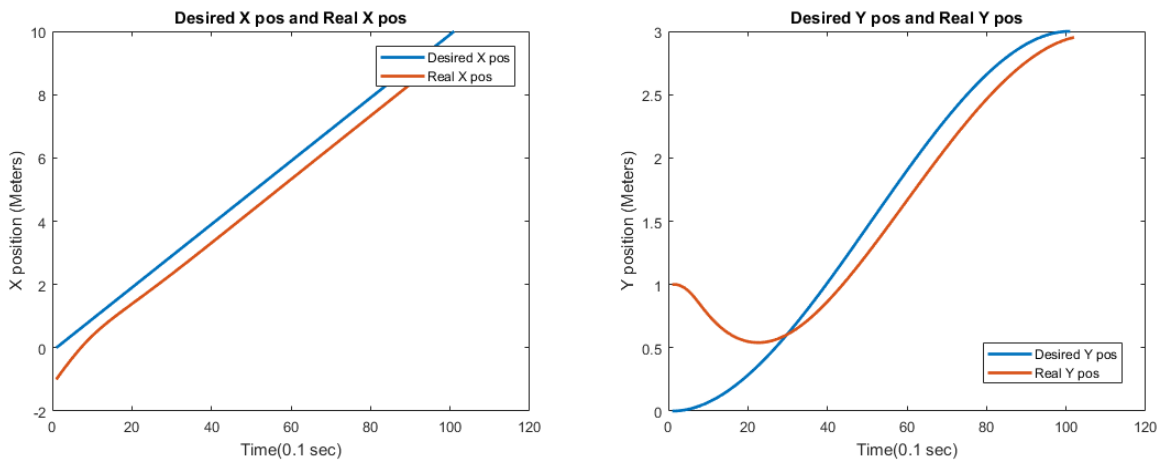
Figure 27: Left – Simulation of x Position

Right – Simulation of y Position

In Figure 27, there is also a consistent offset here. The plot of y positions shows that the controller only needs less than 2 seconds to quickly find the correct direction and position to start following the desired path. Below are the plots of linear and angular velocities of simulated car and desired trajectory:
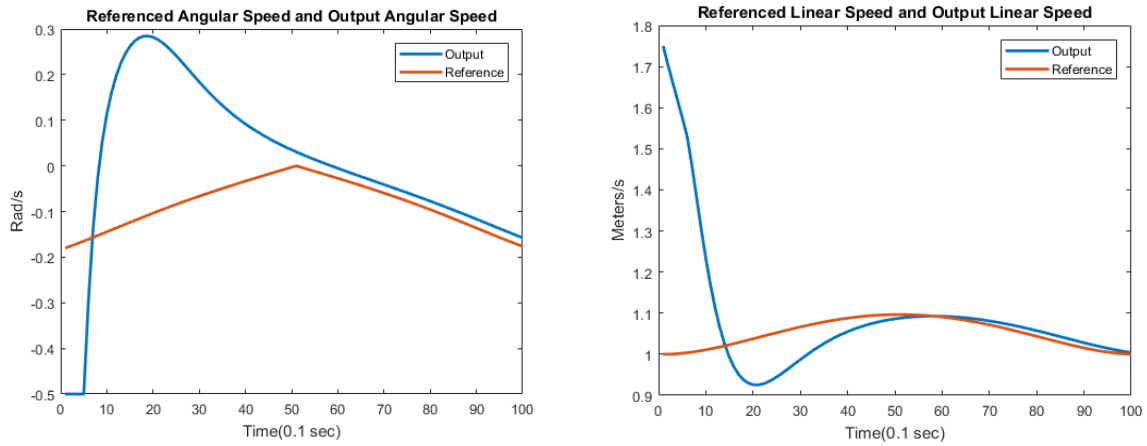
Figure 28: Left – Simulation of Angular Speed

Right – Simulation of Linear Speed

Similar to Figure 25, the linear and angular velocities when the car starts at different position also try to approach the desired velocities and stick closely to the desired velocities. The trajectory shows that even the car is very far from the trajectory starting point, the car still able to run very smoothly and the error did not grow with the time increase.

Since we expected the sensor for localization is not perfectly reliable, we test our controller with add-on random noise from -0.5m to +0.5m.
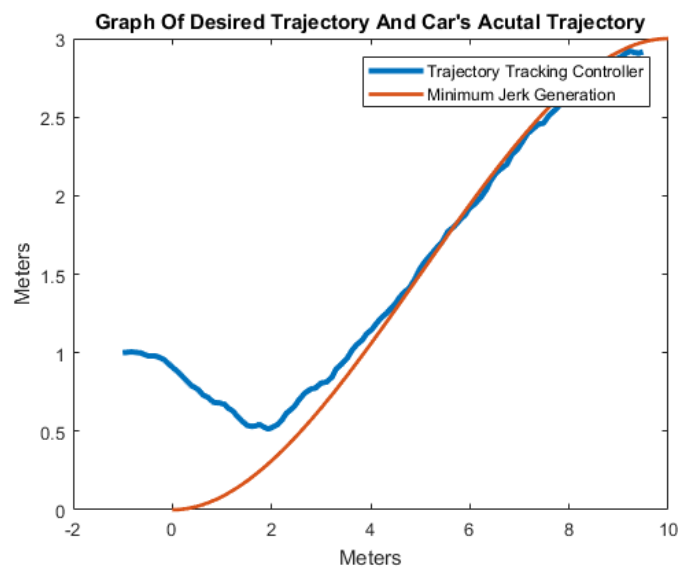
Figure 29: Simulation of Desired vs Actual Trajectory at Different Starting Point with Noises

As shown in Figure 29, even with unreliable localization, the controller still converges. The error would not accumulate.

Figure 30 shows that the trajectory generated by the controller converges after around 2.5 seconds and follows the desired path very well. The random noises resemble the lightings, sounds, and other unexpected factors when real-life sensors work. The noises can also represent the errors from using rounded numbers in the localization algorithm. Despite all that, the controller's trajectory stll turns out well. The offset error from the noises do not accumulate and overturn the output completely. Below are the plots of x and y positions of the car compared to the desired positions:
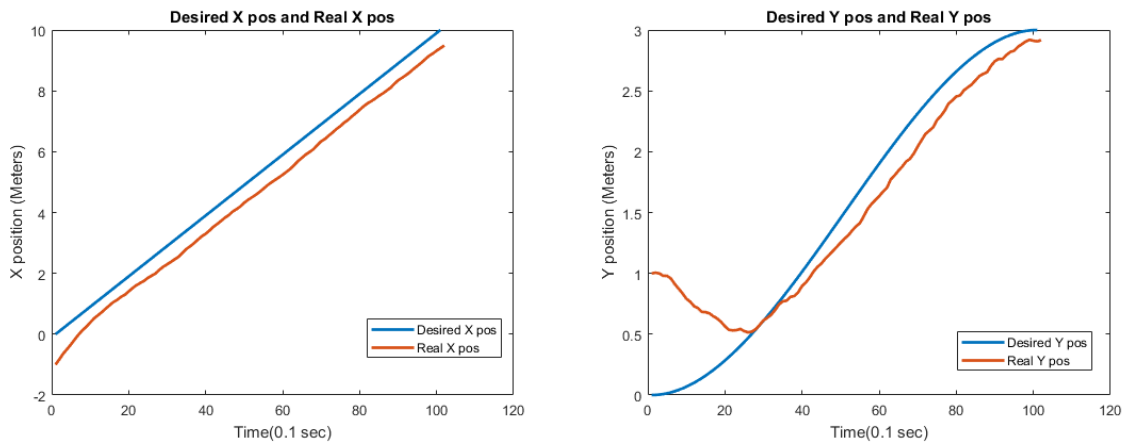
Figure 30: Left – Simulation of x Position

Right – Simulation of y Position

In Figure 30, the x and y positions curves quickly converge and resemble the desired curves. The random noises do not affect much. When applied to real-life system, this can yield a satisfactory results as hardwares and firmwares often cause incompatibilities with each other and other technical issues that can shift the big picture of the project.

The result clearly shows that the car still able to track the trajectory very well and generally smooth.

We also tested some extreme condition with the car start from different angle from the trajectory starting point. Figure 31 shows the result of the controller when the car faces 90 degrees away from the starting angle.
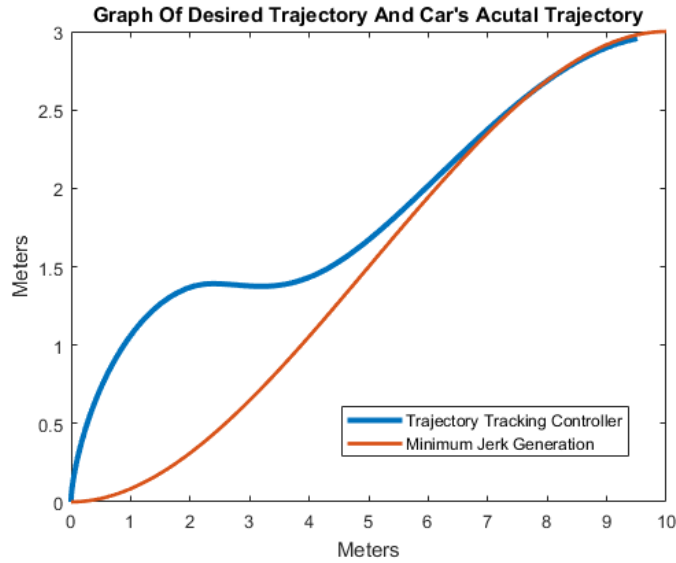
Figure 31: Simulation of Desired vs Actual Trajectory with the Car Facing 90° Away From the Goal

Even though it still converges to the desired path and reaches the goal in the end, the trajectory generated by the controller still needs longer time in-between. This is because the car has to start at 90 degree away from the expected start point. Taking this in to consideration, converging in around 6 seconds is a remarkable speed. Below are the plots of x and y positions of the car compared to the desired positions:
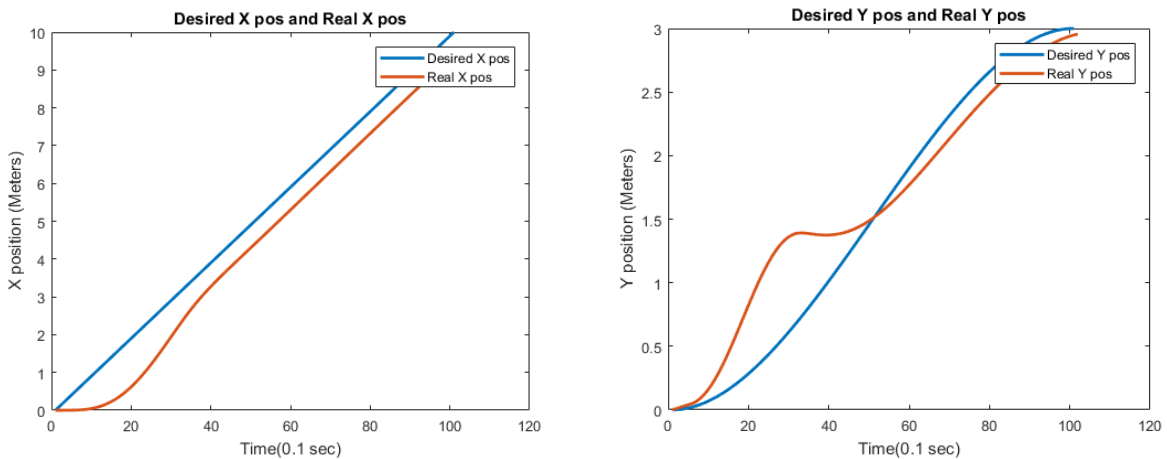
Figure 32: Left – Simulation of x Position

Right – Simulation of y Position

Figure 31 shows the struggle of the controller when it tries hard to converge. The x position curve has always be a straight line from the starting point. But here it take more efforts to be straight. The y position curve has to make a drastic turn in order to take the shape of the desired curve.
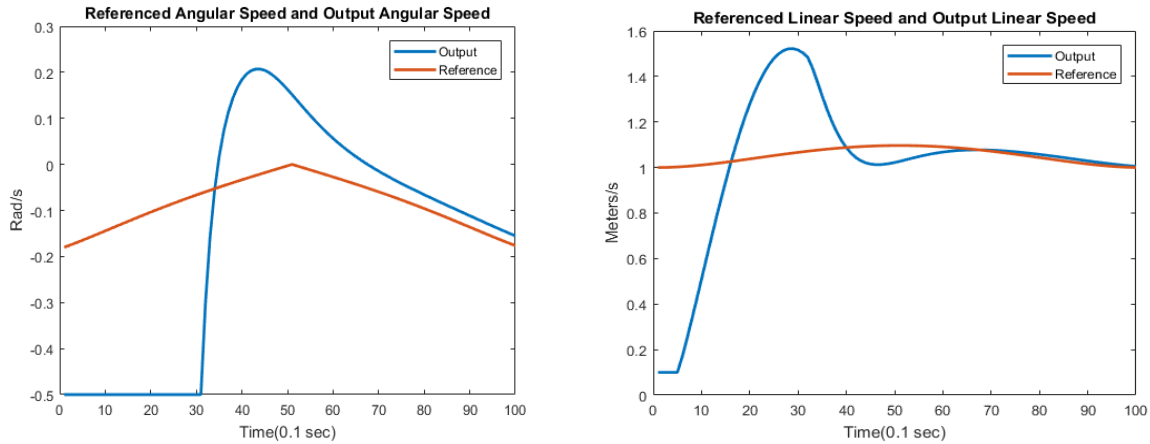
Figure 33: Left – Simulation of Angular Speed

Right – Simulation of Linear Speed

In Figure 33, the linear velocity goes overshoot when it tries to quickly converges to the desired velocity. The angular velocity is smoother when it tries to converge. This is probably due to the car not able to make sharp turns, hence low angular velocity. The result shows that even the car is a little bit far from the desired trajectory, it still merges to the desired trajectory at about 6 second, and track the rest part of trajectory very well.

Figure 34 shows the result when the car faces 180 degrees away from the trajectory starting direction:
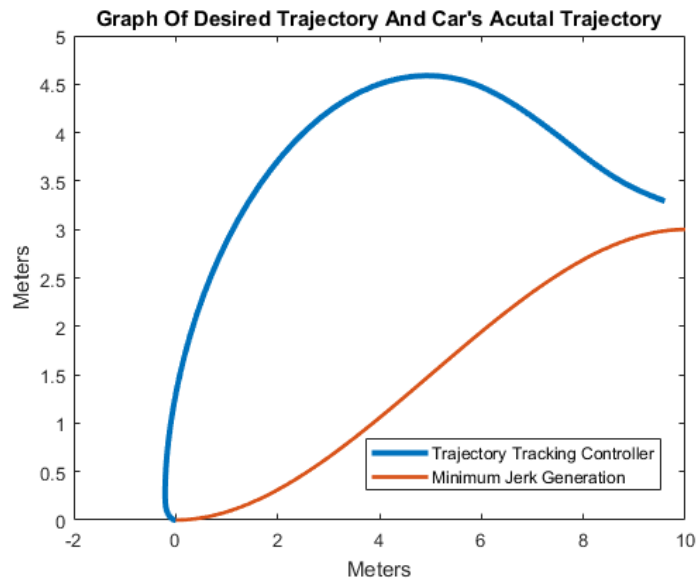
Figure 34: Simulation of Desired vs Actual Trajectory with the Car Facing 180º Away From the Goal

Even thought the trajectory does not converge, the end point is very close to the desired end point. In fact, if the controller is given more time to simulate over a longer distance, the trajectory would eventually converge with the desired path. Compared to Figure 31, the controller has to double the efforts in order to compute the outputs every time it reachs a waypoint.
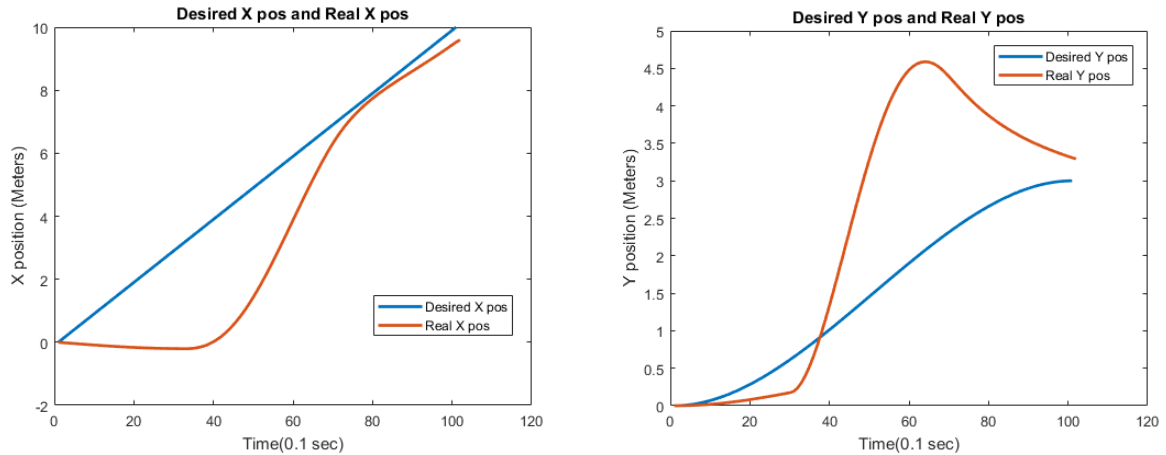
Figure 35: Left – Simulation of x Position

Right – Simulation of y Position

In Figure 35, the x position curve succcessfully matches with the desired curve at the end, thought not accurately, the y position is not able to get close to the desired curve at the beginning, due to the starting angle.
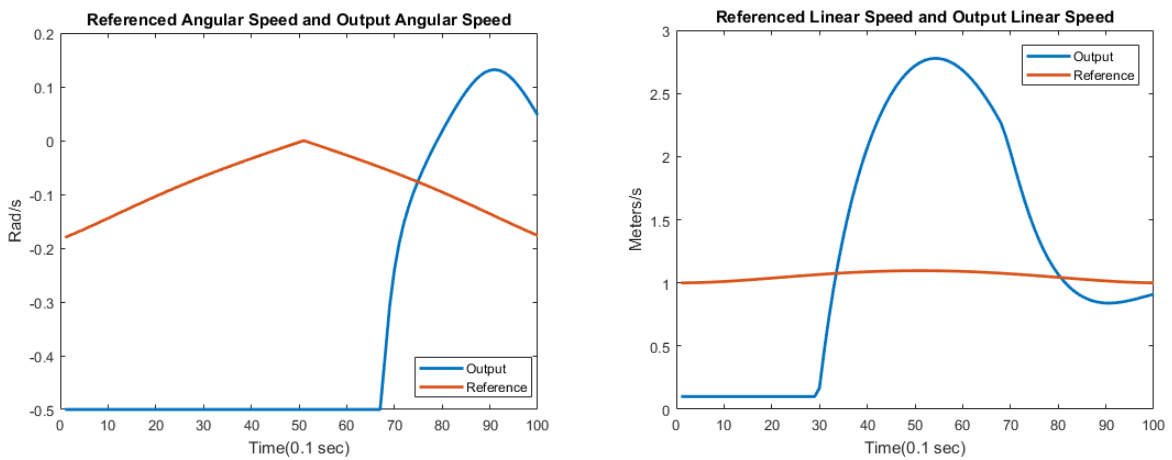
Figure 36: Left – Simulation of Angular Speed

Right – Simulation of Linear Speed

As can be seen in Figure 36, the angular and linear velocity successfully got close to the desired curve. The controller has to drive away from the desired trajectory at first to make a 180 degree turn. If given more time, it will surely converge with the desired curve.

From the result we got from simulation we can get the conclusion that, whether the car start with the same point as the trajectory the car's trajectory will converge to desired trajectory very fast and very smooth. The controller is very robust.

We also tested the controller with different saturated inputs. Figure 37 shows that different saturate inputs create different speed limits. The orange output does not reach the limit yet, hence its peak. The other two, however, have their outputs capped at 0.3 and 0.2, respectively, since their outputs exceed the limits.In return, the two outputs retain high speed output for longer time. This is because it takes more time for the car to reach the waypoint if it drives at low speed.
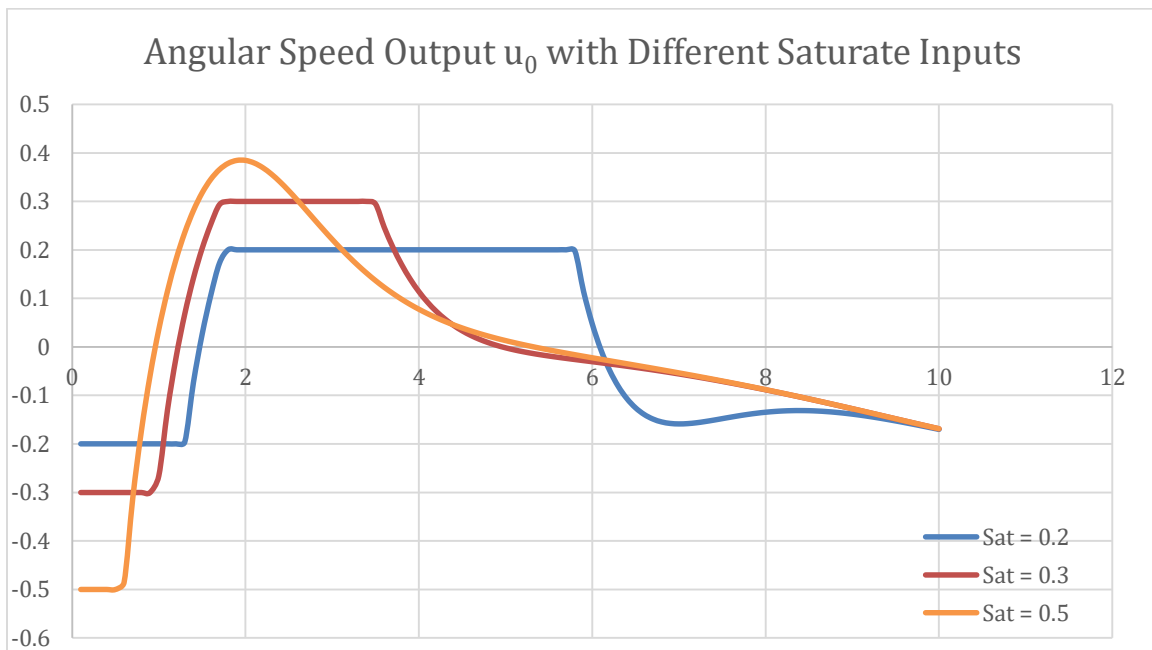
Figure 37: Plot of Angular Speed Output with Different Saturate Inputs

Figure 38 shows how the trajectory generated by the controller responds accordingly to the changes in the inputs. The blue trajectory corresponds to the slowest velocity output in Figure 37. It is also the slowest to converge but the smoothest. The orange trajectory makes the sharpest turn, which is

fairly difficult for our RC car and also dangerous in real life. But it takes the least amount of time to converge.
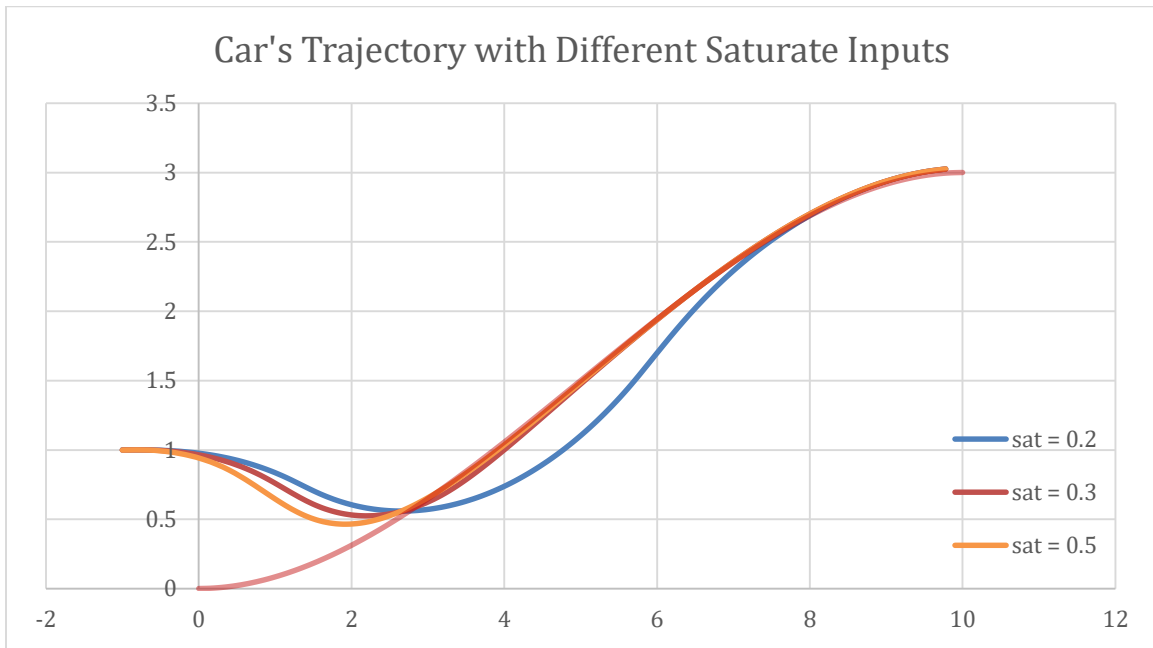
Figure 38: Plot of Trajectory with Different Saturate Inputs

All the outputs are strictly bounded by the saturated inputs. The trajectory will be smoother when the saturated inputs getting smaller, but it will take longer to merge to the reference trajectory.

5.2.2.2   Real World Testing Result

After all the simulation, we tested this controller on our RC car. We test its performance under VICON motion capture system and visualize the behavior in RViz. We use the starting position of the car as the starting point of trajectory and manually give the goal and let the car generate the trajectory automatically.
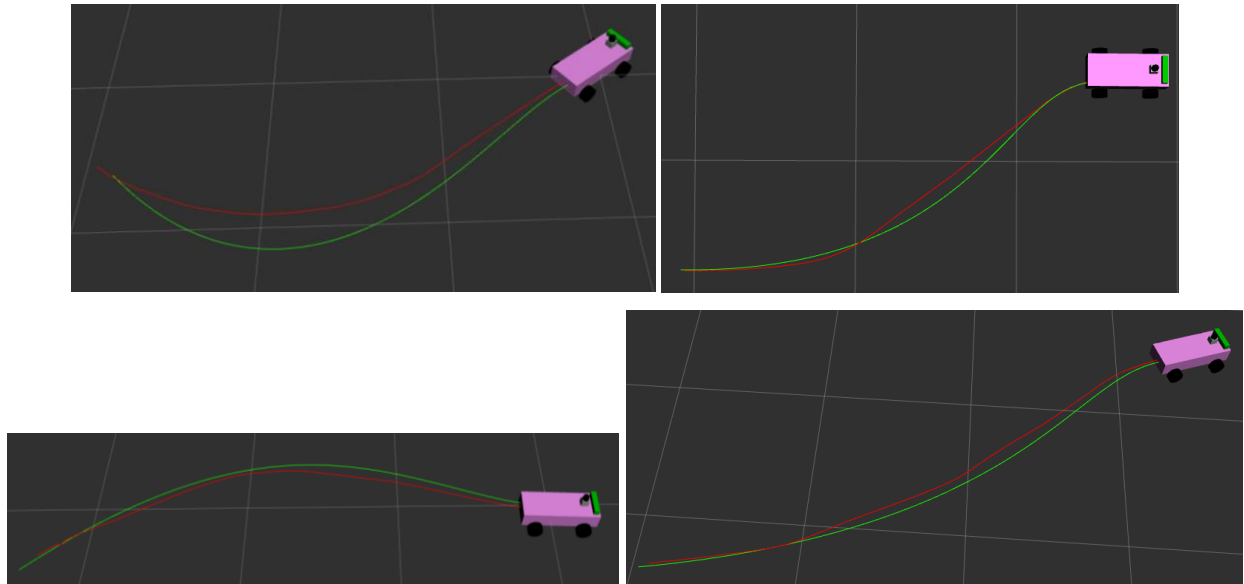
Figure 39: RViz Captures of Actual Trajectory (Red) vs Desired Trajectory (Green)

The green line is the desired trajectory generated by minimum jerk trajectory and red line are the car's actual trail. The test above shows the car can generate a smooth trajectory and follow it smoothly with very small error. We optimize the algorithm and found the car has the best performance when the whole system runs in 10 Hz. Since both of our localization cannot provide a stable and exact coordinates, we change the ending condition and made the car stop when it reaches the point that close to the desired goal and the angle is not much different from the desired pointing angle. After optimizing, we made the absolute error smaller than 10 cm and angle smaller than 10 degrees, which is also the ending condition.

5.3 CVT Performance
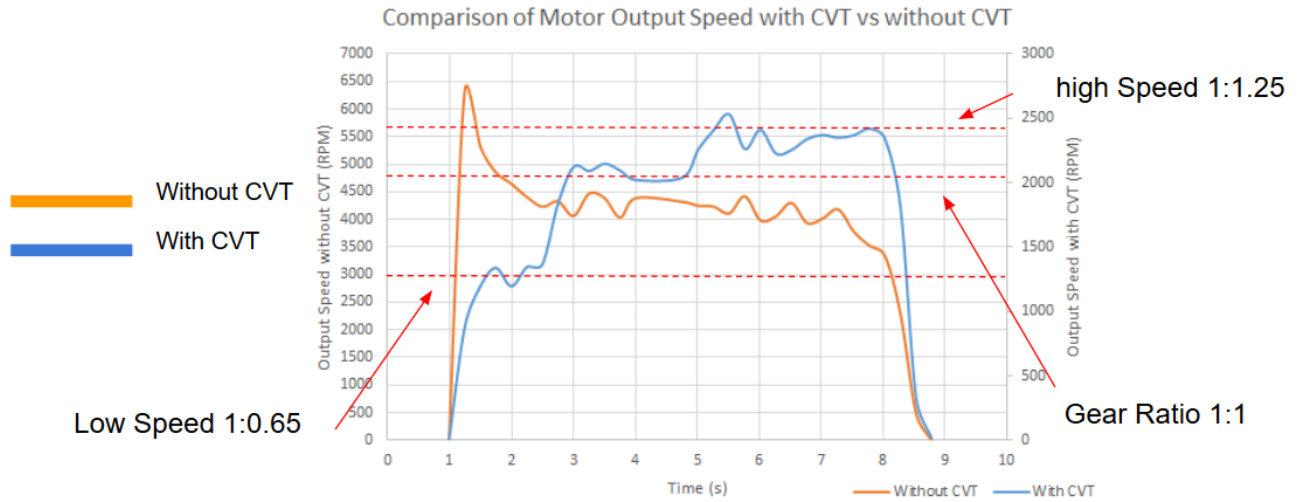
Figure 40: Plot of Motor Output with vs without CVT Attached

We measured Rpm of input and output by optical shaft encoder connected with Arduino. We gain our results in Figure 40 from encoder by Time and RPM, then plot them in Microsoft excel. As we can see in the graph, Orange line define motor original performance without CVT. Acceleration directly goes to 6500 RPM in 0.2 secs approximately. This 6500 RPM is the minimum RPM input of our Traxxas High speed motor. The blue Line indicate the performance with CVT, when in Normal condition, Gear ratio 1:1, our RPM is 2000 RPM, in high gear, 2500 RPM, in low gear, 1300RPM. With our CVT, our acceleration and deceleration are smoothly and in safe conditions. Most importantly, there will be no spike when directly input motor speed.

6 Conclusion and Future Plans

6.1 Discussion

While theoretical researches and mathematical models are abundant, resources about concrete implementation are scarce and specific to various platforms. To bridge the gap between theories and real world applications can be complicated. An error can be raised from anywhere of the system, small as bad wire connection, or malfunction parts, they all can be difficult to debug.

6.1.1 Localization

Gmapping was one of the most commonly used methods to generate a map under ROS environment. However, we encountered some problems when using it. As shown in the previous section, the map we generated has some noises that were mistakenly considered as obstacles. This was due to the odometry data generated from ZED camera. Odometry from encoder would be more reliable compare with those from ZED.

Vicon system generated very precise result compare to the first method we first implemented. It was not necessary to generate a map since the cameras only tracks the markers. The disadvantage with it was the testing filed being too small. With the physical limitations, the car could not follow a very jerky trajectory, which constrains the shape of testing trajectories. Using Vicon system also meant we could not test outdoor.

6.1.2 Adaptive Cruise Control

From the result, we can clearly find out that as long as the front distance is correctly provided by LiDAR, and the driving system is fully functional, the whole system will be very stable and robust. Even though, the safety is mathematically proved, this algorithm is still limited in some way. This algorithm can reflect and make decision within 0.025 seconds but it still cannot handle the lock-braking problem. An ABS (Anti-lock Braking System) should also applied to guarantee the braking efficiency. In addition, while the car is driving with a very high speed and an animal just jump out from the side of the road, the car does can make the braking decision very fast, but it doesn't change the car's actual braking distance.

There were several attributes of the hardware itself that had significant influence on the performance. The first would be the choice of the motor. The driving system of the car was built for racing

purposes which means it could not drive smoothly under low speed. As described in the result section, the original motor would not move if low speed command was received. The way we were testing the algorithm was to have a person walking in front, which would require the car to run relatively slow to show the car can adapt to the person's speed. In the first couple of tests, since the motor was too fast, the car would remain still till the person walks far away then suddenly speed up. Needless to say, the car would often violate the minimum safe distance. Changing to a low speed motor would significantly improve the overall performance.

Braking was also an issue. During the tests, the car could not come to a fully stop when the person suddenly stops in high speed. We were able to change the drag brake constant of the ESC to 90 percent, but we couldn't figure out how to use active brake. The motor and ESC were again built for racing purposes, not for academic studies. The documentation was vague and we didn't get much help from their manufacturer. If other features of the motor would be needed, future MQP teams may want to continue exploring on this.

In this project, we needed a lot of sensors and software packages. Each of them had different data type and protocols. To use them with ROS, proper software needed to be set up. This kind of software was usually written by the manufacture company or package maintainer. Sometimes it could be faulty and made the sensor unusable. The Lidar driver, for example, had some errors with the Ethernet protocol and could not launch the new Lidar properly. We had no choice but to borrow a Lidar of the old model (which uses USB instead of Ethernet) from another MQP team and waited for the maintainer to fix the errors. Two pieces of software that worked fine separately could rise problems when working together if they were not compatible with each other. These could fail the entire project even if the core algorithm works perfectly.

6.1.3 Line Tracking

The overall performance of this method that combines the minimum jerk trajectory generation and trajectory tracking controller using control lyapunov function with saturated inputs is very good. The control lyapunov function provided a converge guaranteed controller and make the whole system can handle very large error and disturbance. However, during the testing we found some part that can be improved with this method.

The first part is the trajectory generation algorithm only take care of smoothness but did not take care of car's physical feasibility. When the starting angle and ending angle have a large difference, the minimum jerk algorithm sometimes might generate a curve with very large yaw rate. However, since the

car has its physical speed and steering limitation, the car's usually need to drive a large circle to reach the desired state.

The second part is the controller cannot handle the situation when it needs to turn from –pi to pi. Normally, when the car is turning, it should be able to rotate to any angle from -180 to 180 freely. However, since we are using controller and matrix to calculate the result mathematically, the controller doesn't know -180 degrees and 180 degrees are actually the same angle. When the reference trajectory need the car drive from -179 degrees to 179 degrees, the car only need to turn 2 degrees, but actually the car will turn 358 degrees, all the way from -179 to +179.

The third part is the car right now can only follow a list of waypoints but cannot adapt the speed of the car flow. It cannot adapt its speed to the front car's speed and cannot brake when front car brakes.

For the first part, we found a trajectory generation method called minimum snap trajectory generation. Snap is the derivative of the jerk. It needs to use optimize function and inequalities to constraint the trajectory and make it feasible for the car's physical structure. For the second part, we think that we can transform the position from global coordinates to car's coordinates. Therefore, the car can never reach its blind point since it will always point to rear of the car. For the third part, since both ACC and trajectory tracking controller are based on control lyapunov function, the objective function in ACC can be replaced by the objective function implemented in trajectory tracking controller. Therefore, the car can not only follow a non-straight trajectory but also have barrier functions that provide hard constraints to guarantee safety. Unfortunately, this is only a one-year project and we can only focus on some specific parts using our limited time.

6.1.4 Half Toroidal Continuously Variable Transmission

our results highly prove that our CVT can provide smooth acceleration and deceleration, there are still some problems we need to consider. First, due to manufacture tolerance issue, we cannot have very accurate and precise dimensions of all the roller and disc that we need. This will directly lead to the vibration of whole system. Secondly, our connection of encoder between output, input and motor are not precise as well. There are more vibrations The encoder we use sometimes cannot tolerate that high speed of RPM, if we can use more precise encoder we can have better results apparently. The total energy loss through the whole system is around 50% due to vibrations and encoder accuracy problems.

Our material is aluminum and acrylic. Those material are not robust enough. In the future, we can change the input, output and power roller to stronger material, such as cast iron or alloy to have more accurate and precise results. The trunnion on the top we are using vex gear, gear ratio is 12:36, but due

the loose fit on the bottom of trunnion and acrylic material, the motion of turning is not smooth enough. If we can change gear to metal and switch the box material and trunnion to cast iron and aluminum, smooth motion of turning there will be. More importantly our vibration of whole system will be less.

6.2 Possible Future Projects

GPS based localization might be very useful to test the car outdoor. Wireless communications are needed to send command to the car. Router and batteries to power it could be one solution. Since GPS sensors usually have low precision, local localization could be helpful.

It would be very exciting to see the car driving around the campus. Computer vison with extended Kalman filter would help with localizing the car in known map. With the algorithms we had, future MQP teams could combine ACC and trajectory tracking.

Other motion planning algorithms could also be implemented on the car. If the car can drive outdoor on itself, the data collected could be used for neural network training.

The car was built for racing purposes, which put limitations on customizing it. A platform that better suits our task could save a lot of work on hacking into the system and deal with all the incompatibilities.

The Lidar sensor is very fragile and expensive. A protection mechanism is needed if testing the car in high speed or other risky situations. The Lidar sensor generates a lot of heat while operating. A heat sink or other heat reducing mechanisms are needed if plan to operate the Lidar for a long time.

Zed camera needs a large USB bandwidth to achieve its highest performance. Even though it was connected with a USB 3.0 port, the Zed camera could not operate under its optimal performance. Improvements could help to make the best use of Zed camera.

6.3 Conclusion

The goal of this project was to implement self-navigating and speed-control algorithms on the car and to build a prototype of the half-toroidal continuously variable transmission. In order to accomplish this, we researched the radio-control car, microcontroller units, processor platforms, sensors, and other related hardware in order to get the car run autonomously without any expectations. Then we researched lane keeping algorithm to get the car run parallel to a wall or a line. We researched ACC to get the car speed up if there are no obstacles ahead or slow down if there are. We researched localization to make the car be able to map an area and be aware of its position on the map. We researched trajectory tracking controller to make the car be able to plan the motions ahead on the map and follow them. We researched obstacle avoidance to get the car avoid obstacles on its path and resume the trajectory. We research CVT to make the car be able to drive normally at low speed. With these ideas in mind, we designed a project that challenged us academically but also offers another view on the current situation of the lack of open-source autonomous vehicle projects. Over the course of one academic year and a summer, the team went from ideas to reality, learning and solving issues along the way.

We were able to successfully get the autonomous car up and running, to implement ACC, trajectory tracking controller, and to build a prototype of the CVT. We fulfilled all the basic goals except for driving straight. We were not able to reach the desired goals, unfortunately, due to unforeseen issues with hardware and software packages.

This project has great potential to develop into a full-fledged development of a racer car that can drive at high speed competitively. With all the goals completed, we believe that the car can drive normally in a highway with aggressive or cautious styles or join a racing competitions. All in all, we have successfully laid a foundation system on the car for the future teams to test their own methods and algorithms, to expand on the goals of our project, and to explore the nature of the autonomous vehicles.

7 Reference

[1]   A. Davise, "The oral History of the Darpa Challenge, the Grueling Robot Race That Launched the Self_Driving Car," CNMN Collection, 03 08 2017. [Online]. Available: https://www.wired.com/story/darpa-grand-challenge-2004-oral-history/. [Accessed 20 04 2018].

[2]   C. Thompson, "These images show how far self-driving cars have come in a few short years," Business Insider, 22 Oct 2015. [Online]. Available: http://www.businessinsider.com/the-first-self-driving-cars-that-competed-in-darpa-grand-challenge-2015-10. [Accessed 20 04 2018].

[3]   "Initiative on Cities and Autonomous Vehicles," Bloomberg Philanthropies, 2017. [Online]. Available: https://avsincities.bloomberg.org/global-atlas/about. [Accessed 12 04 2018].

[4]   "Driverless car market watch," 2017. [Online]. Available: http://www.driverless-future.com/?page_id=384. [Accessed 2018].

[5]   O. Garret, "10 Million Self-Driving Cars Will Hit The Road By 2020 -- Here's How To Profit," Forbes, 03 Mar 2017. [Online]. Available: https://www.forbes.com/sites/oliviergarret/2017/03/03/10-million-self-driving-cars-will-hit-the-road-by-2020-heres-how-to-profit/#451c60fa7e50. [Accessed 01 04 2018].

[6]   C. A. (. 2), F1/10, [Online]. Available: http://f1tenth.org/car-assembly. [Accessed 2018].

[7]   L. Barack, "Just how dangerous are autonomous cars?," Gearbrain, 26 Jan 2018. [Online]. Available: https://www.gearbrain.com/autonomous-self-driving-cars-dangerous-2528732210.html. [Accessed 2018].

[8]   "Learn about autonomous cars.," 30 Jan 2015. [Online]. Available: http://www.openroboethics.org/autonomous-cars/. [Accessed 2018].

[9]   "Autonomous Cars Survey Responses," [Online]. Available: https://docs.google.com/forms/d/e/1FAIpQLSf1f_8NOYvp22UgyXLGMxRLXo0jfSIF1HQT7y9U2lsoTbc_kg/viewanalytics?embedded=true.

[10]   Kangalow, "Jetson Based Autonomous Race Car – University of Pennsylvania," 13 June 2016. [Online]. Available: http://www.jetsonhacks.com/2016/06/13/jetson-based-autonomous-race-car-university-pennsylvania/. [Accessed 2018].

[11]   T. I. Daping Liu, "The Latest Technology of Traction Drive Half-Toroidal CVT," in *Advanced Tribology*, Springer, Berlin, Heidelberg, 2008, pp. 930-931.

[12]   L. NISSAN MOTOR CO., "EXTROID CVT For Application to Rear-Wheel-Drive Cars Powered by Large Engines," Nissan's CVT Technologies.

[13]   "2015 Motor Vehicle Crashes: Overview," US Department of Trahsportation, August 2016. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812318. [Accessed 2018].

[14] "Critical Reasons for Crashes Investigated in the," US Department of Transportation, March 2018. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812506. [Accessed 2018].

[15] H. R. A. M. Sangjun Park, "Fuel Economy Impacts of Manual, Conventional Cruise Control, and Predictive Eco-Cruise Control Driving," *International Journal of Transportation Science and Technology,* no. 10.1260/2046-0430.2.3.227, pp. 227-242, 2013.

[16] "A comprehensive review of the development of adaptive cruise control systems," *International Journal of Vehicle Mechanics and Mobility,* vol. 48, no. 10, pp. 1167-1192, 2010.

[17] A. E. Bryson, "Applied Optimal Control: Optimization, Estimation, and Control," Washington: Hemisphere Pub. Corp.; New York: Distributed by Halsted Press, 1975, p. 481.

[18] J. P. R.A. Freeman, "Control Lyapunov functions: new ideas from an old source," in *Decision and Control, 1996, Proceedings of the 35th IEEE Conference on*, 1996.

[19] J. A. P. V. N. J. C. Doyle, "NONLINEAR OPTIMAL CONTROL: A CONTROL LYAPUNOV FUNCTION AND RECEDING HORIZON PERSPECTIVE," *Asian Journal of Control,* vol. 1, no. 1, pp. 14-24, 2008.

[20] D. Panagou, D. M. Stipanovic and P. G. Voulgaris, "Multi-objective control for multi-agent systems using Lyapunov-like barrier functions," in *52nd IEEE Conference on Decision and Control*, 2013.

[21] M. Rauscher, M. Kimmel and S. Hirche, "Constrained robot control using control barrier functions," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 279-285, 2016.

[22] "Control Barrier Function based Quadratic Programs with Application to Adaptive Cruise Control," *Proceedings of the IEEE Conference on Decision and Control,* vol. 2015, no. Feburary, pp. 6271-6278, 2015.

[23] Y. X. a. B. L. Panfeng Huang, "Global Minimum-Jerk Trajectory Planning of Space Manipulator," *International Journal of Control, Automation, and Systems,* vol. 4, no. 4, pp. 405-413, 2006.

[24] R. B. Wei Ren, "Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints," *IEEE Transactions on Control Systems Technology,* vol. 12, no. 5, pp. 706-716, 2004.

[25] "Overview of materials for Acrylonitrile Butadiene Styrene (ABS), Extruded," [Online]. Available: http://www.matweb.com/search/DataSheet.aspx?MatGUID=3a8afcddac864d4b8f58d405 70d2e5aa&ckck=1. [Accessed 2018].

[26] "Geometrical optimization of half toroidal continuously variable transmission using particle swarm optimization," *Scientia Iranica. Transaction B, Mechanical Engineering,* vol. 18, no. 5, p. 1126, 2011.

[27] "Electric Motors - Power and Torque vs. Speed," The Engineering ToolBox, [Online]. Available: https://www.engineeringtoolbox.com/electrical-motors-hp-torque-rpm-d_1503.html. [Accessed 2018].

[28] "Friction and Friction Coefficients," The Engineering ToolBox, [Online]. Available: https://www.engineeringtoolbox.com/friction-coefficients-d_778.html. [Accessed 2018].

[29] "TRAXXAS VELINEON VXL-3S WATERPROOF BRUSHLESS POWER SYSTEM, 3350R," RC Hobby Explosion, [Online]. Available: https://www.rchobbyexplosion.com/Velineon_Waterproof_Brushless_System_p/tra3350r.htm. [Accessed 2018].

[30] E. &. J. F. D. (. Oberg, Machinery's handbook, 16 ed., New York: Industrial Press, 1959.

[31] W. Ren and R. Beard, "Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints," *IEEE Transactions on Control Systems Technology,* vol. 12, no. 5, pp. 706-716, 2004.

8 Appendix

8.1 Appendix A: Bill of Materials

- Lab property
    - Lidar (1700)
    - Zed (500)
    - RC car (400)
    - Tx2 (400)
    - Power Bank (150)
- MQP Budget
    - Teensy (25*4)
    - USB Hub (40)
    - Cable (50)
    - SD card (40)
    - RPM sensor (20)
    - Rope (25*2)
    - CVT material (100)
    - Power Bank (150)
    - CVT manufacture (150)
    - New Motor (200)

8.2 Appendix H: Timeline

Planning / ACC / Lane Keeping

	Phase	Task
A	Planning	1 Proposal
		2 Decide the time line
		3 Re-calibrate PID
	ACC	4 Setup Lidar
		5 Perfect ACC
	Lane Keeping	6 Make the testing ground
		7 Start debugging LK

Lane Keeping / Lane Changing

	Phase	Task
B	Lane Keeping	1 Test with current LK algorithm
		2 Try a different algorithm (steering control only)
		3 Compare between two
		4 Calibrate the better one
		5 Finishing up the algorithm
	Lane Changing	6 Start Coding for LC
		7 Coding for LC

Lane Changing / Extra time / Final Report

	Phase	Task
C	Lane Changing	1 Coding for LC
		2 Radar checking system for LC
		3 Debugging LC
		4 Calibrate LC
		5 Test LC with LK
	Extra time	6 Extra time for unknown issue
	Final Report	7 Starting Final Paper

Final Paper / Extra time and submit

	Phase	Task
D	Final Paper	1 Start Introduction
		2 Background research
		3 Methodology
		4 Methodology
	Extra time and submit	5 Finishing up
		6 Submit
		7

Group	#	Task
A	1	Proposal
A	2	design CVT in solidworks
A	3	design and stimulate CVT
A	4	calulate and stumulate in solidworks
A	5	perfect design and stimualtion
A	6	build prototype
A	7	build and test prototype
B	1	design control system
B	2	continous design
B	3	try to order or build control system
B	4	debug for control system and try to build on prototype
B	5	testing control system on prototype
B	6	debug and test system
B	7	continous testing
C	1	debug and perfect protype
C	2	design CVT for final work
C	3	find factory
C	4	order and wait for build
C	5	extra time for waiting
C	6	Extra time for unknown issue
C	7	test final version
D	1	Start Introduction
D	2	Background research
D	3	Methodology
D	4	Methodology
D	5	Finishing up
D	6	Submit
D	7	

8.3 Appendix G: CAD Drawing

Half Toroidal Continuously Variable Transmission CAD FINAL DRAWING

TRUNNION
FINAL

Ø0.43
Ø0.20
Ø0.43
1.48
0.81
0.59

FINAL ASSEMBLY
FINAL

2.63
2.90
0.17
2.15

Ø0.24

0.16

0.16

0.08

R0.53

0.30

	NAME	SIGNATURE	DATE			TITLE:		
DRAWN								
CHK'D						Output Disc		
APPV'D								
MFG								
Q.A				MATERIAL:		DWG NO.		A4
				Aluminium		Orginal		

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE INCHES

FINISH:

DEBURR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING   REVISION

WEIGHT: SCALE:1:1   SHEET 1 OF 1

Drawing 1 — Title block:

	NAME	SIGNATURE	DATE		TITLE:	
DRAWN					**Input Disc**	
CHK'D						
APPV'D						
MFG						
Q.A						

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION

MATERIAL: Aluminium T6061
DWG NO. **Orginal**
A4
WEIGHT:
SCALE:1:1
SHEET 1 OF 1

Dimensions: ⌀1.65, 0.16, R0.53, 1.18, ⌀0.24, 0.08, 0.21

Drawing 2 — Title block:

	NAME	SIGNATURE	DATE		TITLE:	
DRAWN					**POWERROLLER**	
CHK'D						
APPV'D						
MFG						
Q.A						

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION

MATERIAL: Aluminium
DWG NO. **Orginal**
A4
WEIGHT:
SCALE:1:1
SHEET 1 OF 1

Dimensions: ⌀0.95, 0.59, R0.53, 0.20, ⌀0.39, ⌀0.20, ⌀0.74, 0.47

D

0.16

R0.12

C

0.79

Φ0.20

0.31

B

	NAME	SIGNATURE	DATE			
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES			FINISH:		DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
DRAWN						
CHK'D					TITLE:	
APPV'D						Output Disc
MFG						
Q.A			MATERIAL: Aluminium		DWG NO.	Orginal	A4
			WEIGHT:		SCALE:1:1	SHEET 1 OF 1

A