Autonomous Driving Platform Performance Analysis

Charles R. Rickarby
University of New Hampshire, Durham, chrickarby@icloud.com

Autonomous Driving Platform Performance Analysis

Charles Rickarby

Adapted from: Autonomous Vehicle Design Using UNH Driving Simulator

By: Eric Duross, MacKenzie Meyers, Charles Rickarby

ECE Faculty Advisor: Thomas Miller

Abstract:

This thesis was developed in conjunction with a senior project and report pertaining to the development of an autonomous vehicle platform built off of the UNH driving simulator. The goal of that project was to create a control platform that would, through network connections, drive an autonomous car simulator without input from a user. Minor milestone goals of the project were to enable the car to move around a closed loop track without any user input, and enabling the car to go through intersections once the user had decided the desired turning direction via a user interface. The control platform was based off of a third party simulator and a java development environment containing the control code. These two subsystems then communicate information back and forth over a network connection.

This thesis borrows from the background work and research done for that project and expands upon it with performance analysis. In the creation of the thesis it was necessary to develop a method of data collection and interpretation so as to evaluate the performance of the autonomous driving platform developed for the project. This was implemented through the java interface in the form of logging runtime data to a CSV file during each trial of the simulator. These CSV files were then loaded into RStudio and interpreted through various R programming packages to show the performance benefits of the final platform versus some of the tested prototype platforms.

Introduction:

Problem Definition:

For a senior capstone project, continuous development of a control platform for an autonomous car platform will make it possible to study how users interact with autonomous cars and how they react to being in a self driving car. Along with this it will be possible to determine the safety benefits of autonomous cars versus manually driven cars, through tests in a safe, simulated environment.

Furthermore, through data analysis of various plots and figures it will be possible to determine the best control parameters to get the best performance out of the autonomous driving platform. This data, presented in this thesis, will show quantitatively what the best control strategies are through comparison of different versions of the platform.

Simulator Introduction:

The following introductory and background material was developed for the senior capstone project report and adapted for use in this thesis. The development of the control platform took place entirely outside of the body of work pertaining to the thesis, however the information is still useful in understanding the results of this thesis.

The simulator the team is working with is located in Morse hall. It is a sophisticated highly realistic driving simulator (See Figure 2) that includes a world designer (Figure 3) so a user can add traffic patterns, other cars, pedestrians and other environmental factors. These environmental factors such as pedestrians and other cars can be coded to behave in different ways making it possible to test the autonomous car through various real life scenarios. The simulator, when being driven manually, uses USB pedals and a steering wheel to drive the car, and these inputs can be monitored to study driver distraction and driver vehicle interaction. The simulator can also be controlled through java code on a separate computer. The team goal for the capstone project was to build upon the current code base to successfully automate the simulator through different environments and roadways, eventually adding in intersection handling and more sophisticated obstacle avoidance and curve detection.

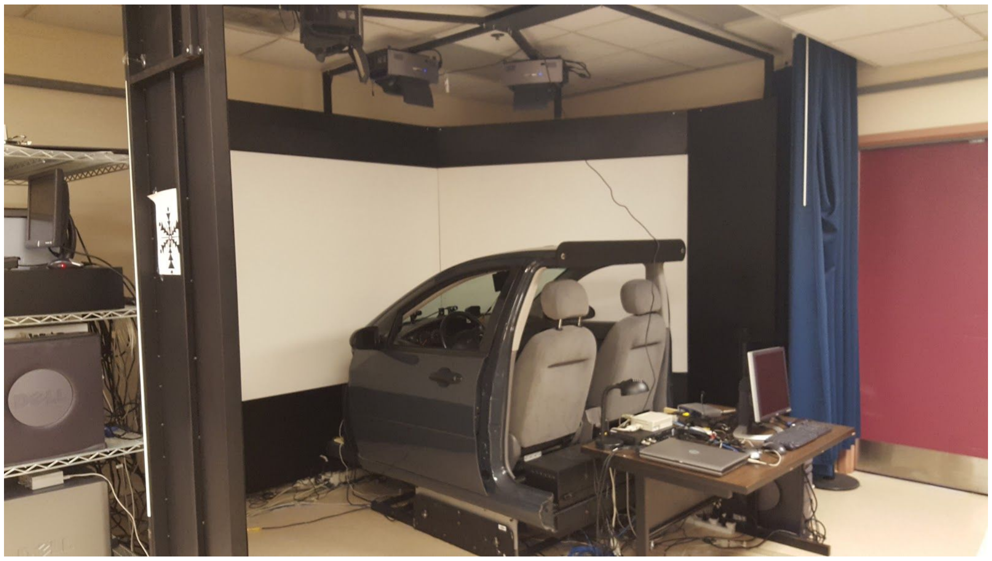


Figure 1: Full size driving simulator

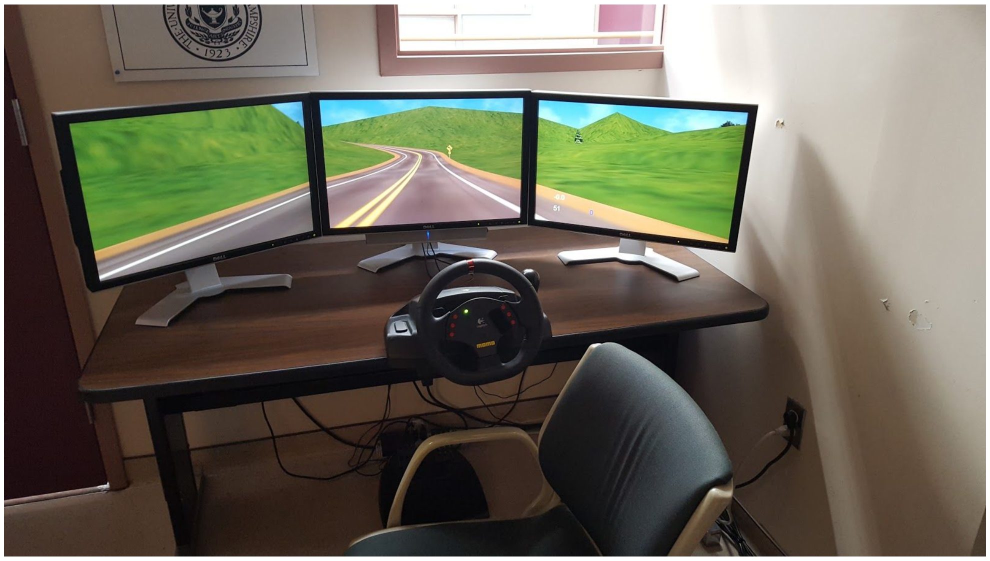


Figure 2: Small scale driving simulator

On a more technical level the simulator is interfaced to a windows computer, and a java application was developed to allow for real time feedback from the simulator and the environment. The autonomous control is being developed in the java environment. The logical block diagram has a more detailed view of the setup of the simulator system.

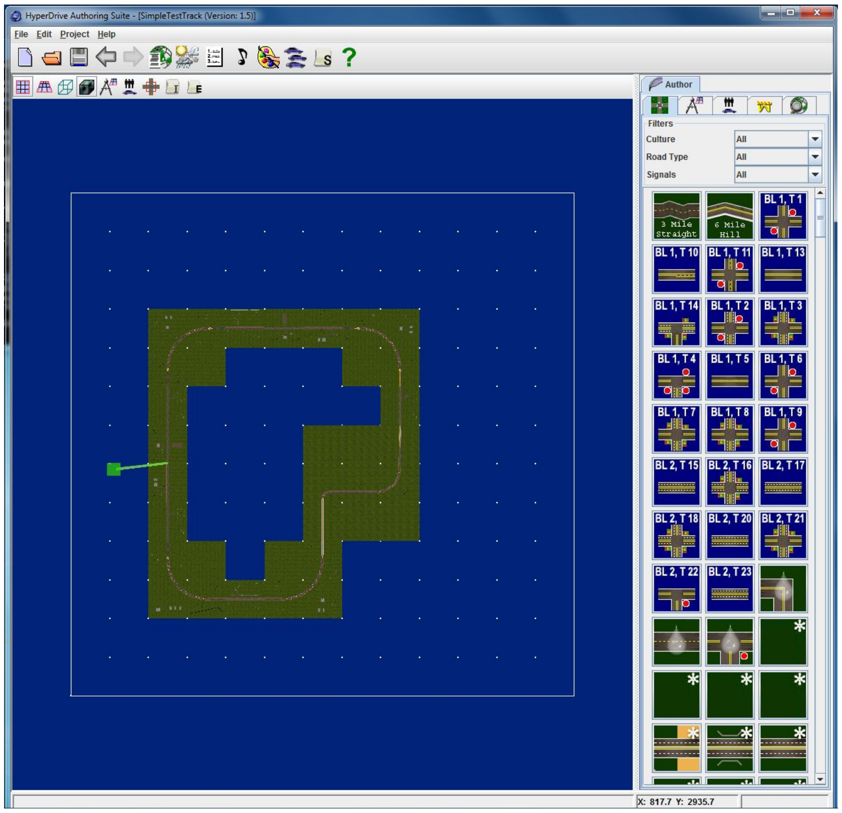


Figure 3: Hyperdrive Authoring Suite: Simulator map designer

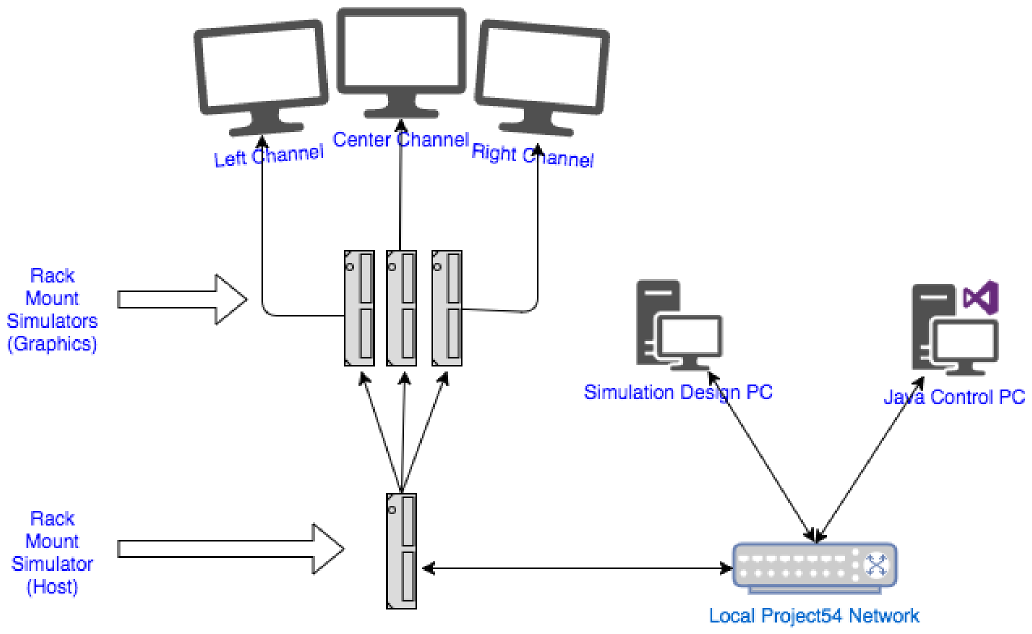


Figure 4: Logical Block Diagram

Design Objectives:

The design objectives of the capstone project were to learn about what sensors are available in real world autonomous cars to ensure that the control platform developed does not deviate from what data would actually be accessible. Also accomplished in the first semester was the development of the control strategies to navigate a closed loop under complete autonomous control. This included speed and steering control while navigating through ambient traffic. This also took into account that the only information that will be used is that which is readily available by real world sensors, and that the control platform will react the same way that real way actuators function. For instance, the car is not able to instantaneously stop or change speed, even though the simulator will allow it, because this is not representative of how a car would actually function in the physical world.

Also implemented is the ability to navigate both a closed loop track and also a track with intersections. Intersections are navigated through a java user interface. This interface allows users to supply directional instructions at intersections as well as display basic data about the car such as the speed. Avoidance of pedestrians and other traffic that deviates from normal traffic patterns is a goal that is currently unimplemented.

Design Methods:

The following design methods are again adapted from the capstone report and do not represent work done solely for the thesis. The design of the autonomous driving platform was completed alongside the thesis, however represents a different body of work. The following information has been included to give background and show significance in the results presented in the thesis.

Implementation Plan:

The implementation plan was to continue on the current design of the self driving car and expand upon its functionality. In the current implementation of autonomous steering has been corrected and developed so that the car will drive in a steadier path at higher speeds. Control of the speed of the vehicle has also been implemented into the autonomous functionality. A user interface has also been implemented to allow the user to select which way the car should go at an intersection.

In future iterations of this project, the autonomous car will be able to follow a car in front of it at a safe distance, and interact safely with ambient traffic. Following that, the car will be able to avoid obstacles like pedestrians.

Testing Plan:

As functionality was added into the vehicle, the testing scenarios changed in order to test the new functionality. To test the improved steering stability, the car was tested on a straight away track. In order to test both speed control and steering capability, the car was tested on a closed loop track with varying features and road conditions. Lastly, to test the user interface for navigation instructions the track was modified to include intersections.

In future iterations of this project, traffic will be added to the closed loop track to test the safe following of a vehicle in front of the car. Pedestrians will also be added to the closed loop track when it is time to test the pedestrian avoidance function.

Team Roles:

- Simulator: MacKenzie Meyers
- Java Interface: Charles Rickarby
- Autonomous Control: Eric Duross

Team roles were flexible and changed on an as needed basis, but certain areas of expertise have been selected to clarify directions and goals as work progresses. The simulator expert was primary responsible for testing and simulation mastery, such as world design to test the driving algorithms and pedestrian and car injection into real world scenarios.

The java interface expert was primarily responsible for the live feedback from the machine to get data and output from the car. This interface functions similarly to a dashboard and display information about the current status of the car, and to handle user input at intersections.

Lastly, the autonomous control expert focused on the algorithms responsible for deciding the car's behavior. Speed and braking changes, automatic turning and turn detection are all covered in this role.

Implementation:

The following implementation information again only pertains to the implementation of the autonomous driving platform and is again included to assist in interpretation of the data presented in this thesis.

Autonomous Control:

First semester developments can mainly be divided into two categories, an improvement on steering control, and a tiered speed control. The steering control improved on that which had already been implemented so that it would be possible for the autonomous car to be more stable at faster speeds. This also improved handling on sharp turns. The original steering control only took into account the current lane position offset from center, and corrected based off of that, attempting to reach a lane offset of zero, which had varying results especially around turns. As this worked somewhat it was decided that before making too many changes to the steering control, a speed control would be implemented.

The steering control structure was such that if the current speed was greater than the speed limit the brake would be applied and vice-versa. The brake and accelerator pedals were then applied on a tiered system based on the difference between the current speed and the desired speed.

Once this simple speed control had been implemented, the steering control was altered to use a “look-ahead” approach. This approach utilizes information from the simulator about what the road looks like ahead of the car’s current position, and using the angle offset between the current heading and the points on the road ahead, would alter the steering wheel angle appropriately. After the steering control was refined using the “look-ahead” approach, the speed control was re-visited using a proportional and differential feedback control structure. By taking into account the previous speed as well as the previous values of the gas and brake pedals it was possible to create more gradual and realistic acceleration and deceleration of the car.

In the second semester, the first development goal was to revisit the steering control to correct and fine tune the controls in order to create a more stable system. Once that had been completed, the next big step in development was the handling and execution of intersections. In order to detect intersections, the number of lanes in the next upcoming lane had to be detected. If this number was greater than one it meant that there was an intersection ahead. To be able to stop at the intersection, the speed limit was multiplied by a factor of the total distance away the stop line was from the beginning of the tile, and how far the car was currently from the stop line. As the car approached the line, the desired of speed of the car went to zero. The final step of development was handling the turns at the intersection. The current approach of look ahead points would not work here, as in the intersection there was no trail of points to the desired exit. The only 2 points that existed were the entrance point to the intersection, and the desired exit point. The algorithm created to handle this took the two points, found the angle between and was scaled by a factor and then set the steering angle to that angle and held it constant throughout the turn. Once the turn was completed, the car reverted back to its autonomous control.

Simulation Design:

The simulator was used to create closed loop tracks of lifelike roadways to properly simulate normal driving conditions. Roadways with more natural curves and corners were selected in road design so that the car could be forced to make use of all design algorithms when implementing the autonomous control. The curvy roads could be combined with inclines and elevation changes. The simulator had limited choices in less random roadways, so to avoid constantly forcing the vehicle through straight tracks and wide 90 degree angles, more natural roadways were selected. To further test the implementation of intersection and braking during the second semester, more urban and intersection heavy maps were used to test the controls. The simulator allowed for scripting of other traffic, pedestrians and even animal obstacles, but real world style obstacle avoidance was never implemented.

The simulator had various limitations despite being an advanced and fairly customizable software tool. The way certain tiles mesh together could be at a slightly unpredictable angle, and the nature of the simulator and autonomous design doesn’t allow for any prediction of an unexpected bad tile connection. This results in the car sometimes having to very quickly correct itself despite appearing like it is about to veer into the other lane or off the road entirely. Another setback, as mentioned before in the intersection implementation section, was that there are no tracking points in an intersection. This means that there was no way to properly track the autonomous vehicle's location while it went through an intersection and completed the turn. This caused some randomness to how the vehicle would handle intersections. This required the team to implement a more accurate and reliable braking method when approaching intersections, so that the turn reliably starts from the same point and the autonomous vehicle can properly detect a roadway following the turn.

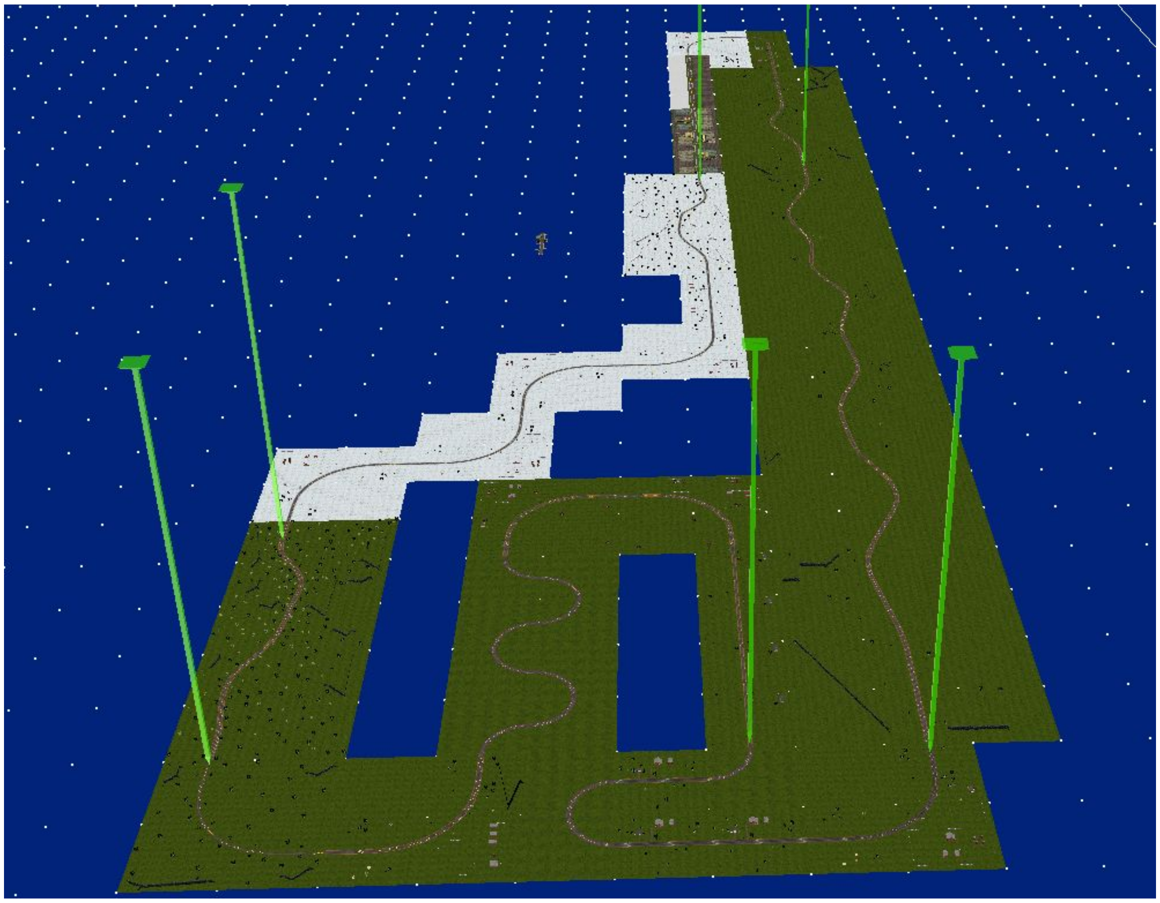
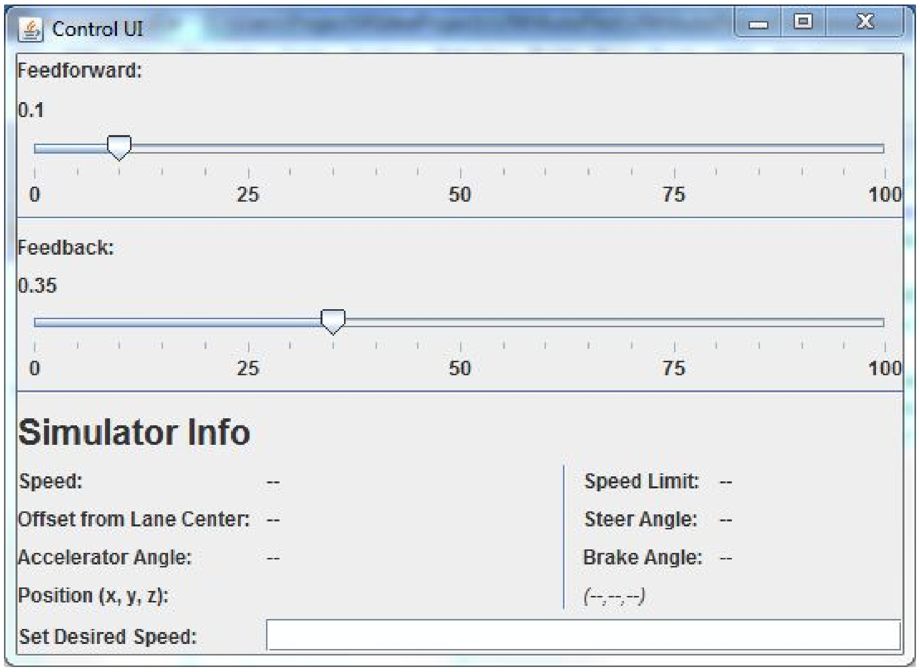


Figure 5: Hyperdrive map example with start points

Above is an example of a closed loop test track like the ones designed for the URC demonstration. The loop above features no intersections so that the autonomous car can constantly run in a loop without the need for user input. These closed loop tracks allowed the team to see that the autonomous car could run independently for very long periods of time without any issue or crashes. The green points are user defined spawn points for easier testing of the car on problematic areas.

User Interface:

The current status of the user interface is split up between in two different application windows, the Control UI and the Navigation UI. The Control UI serves two purposes. The first is to deliver simulator information such as the speed, offset from the lane's center, and the angle of the steering wheel (Figure 6). The second purpose of this interface was to enable some user interaction during the development of the control strategies. This interaction is in two different forms. The first is two sliders that control two variables in the java command code, feedback and feedforward control values. These sliders could be moved in real time to change the feedback control loop of the java code. The two sliders were tied to two global variables in the control code that could be assigned and inserted in different parts of the code to adjust the desired control structure. This would allow the project group to qualitatively determine the best feedforward and feedback values for the simulator and different control structures by watching the simulator and adjusting the simulators if the car was driving poorly.



Feedforward:
0.1

Feedback:
0.35

Simulator Info

Speed:	--	Speed Limit:	--
Offset from Lane Center:	--	Steer Angle:	--
Accelerator Angle:	--	Brake Angle:	--
Position (x, y, z):		(--,--,--)	

Set Desired Speed:

Figure 6: Java Control UI

The other part of the Control UI that helped in adjusting the control algorithm was the ability for the user to set the desired speed at any time while the simulator was running. By allowing this feature it was possible to see how the car would function when the speed was much higher or much lower than the speed limit. This allowed the project team to evaluate the control algorithms effectiveness at different speeds on the same roadway shape and dimension.

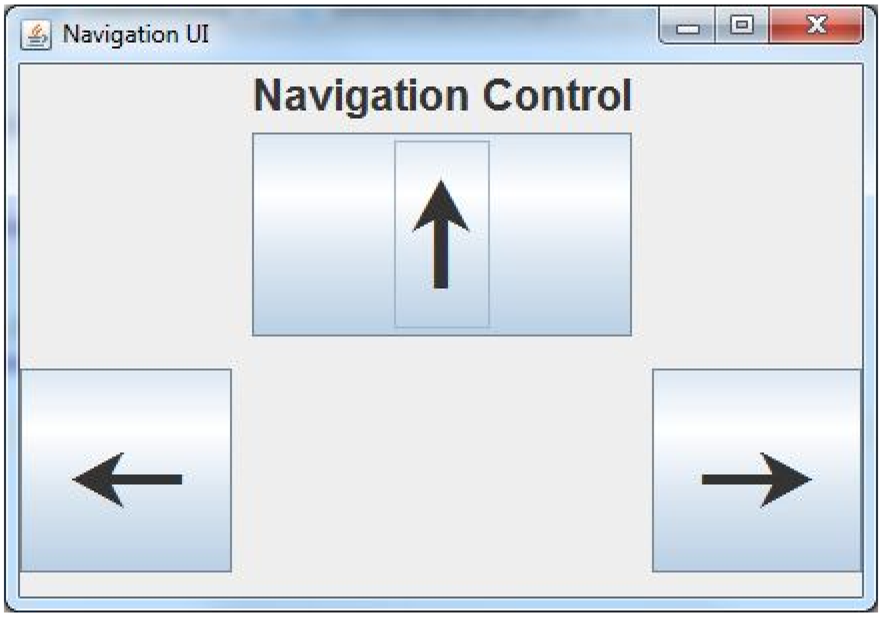


Figure 7: Java Navigation UI

The second part of the java user interface was the Navigation UI (Figure 7). This part of the user interface served the purpose of roadway intersection navigation based on user input. The UI consists of three buttons, left, straight, and right. This very simple UI enabled users to, when the car stops at an intersection, choose the direction that the car would go next when exiting the intersection. Once the user clicked one of the buttons the control loop would restart, repeating a section of the control loop to written to execute a turn in an intersection until the turn completed and the car exited the intersection. At this point the control loop would then exit the intersection handling portion of the code and return to normal functionality.

The UI also extends onto the simulator screens however. The simulator provides the ability to print out values onto the simulator screen in the same way a heads up display would work, and some main variables were chosen to be printed out here. Before the current project team started work on the simulator, already present on channel two, or the right display of the simulator, were the steering wheel angle as well as the offset from the center of the lane. For testing of the autonomous speed and steering controls the brake and accelerator values were added, as well as the current speed and the speed limit of the current road tile. When it came time to implement the feedforward control based on the upcoming points, the car's current x, y, z location was added to the display, as well as the number of "road points" on the current lane, which just describes the shape of the road that the car is on.

Qualitative Results:

The results presented in this section are purely qualitative and adapted from the original project report.

Speed Control:

Of all the control strategies implemented in the autonomous vehicle design platform, speed control is the most reliable. Speed tracking is very effective and operates similar to how a human would drive. The desired speed is based off of the roadway configuration (straight, curve, hill), and the speed limit. There is then a three mile per hour margin of error around that determined desired speed, so the speed fluctuates much how it changes when a human is driving a car.

On straight road sections the speed is uniform, and the desired speed is the same as the speed limit. The car speed fluctuates up and down within the three mile per hour margin of error, however just looking at the vehicle simulation there is no jerkiness and it drives smoothly. On a road that has a curve the desired speed is set lower so that the car handles the turn at an appropriate speed. The sharpness of the curve is also factored into the desired speed. Sharper turns have more simulator road points, and as a result of that the speed algorithm sets the desired speed to be lower than if it had fewer points.

Since the accelerator and brake pedal angles are determined based off of the desired speed set by the speed control algorithm hills both uphill and downhill did not need any extra implementation because the car's only goal is maintaining the desired speed. The pedals will be altered in order to maintain the desired speed (keeping in mind the three mile per hour margin of error.

Steering Control:

The steering control was slightly more involved than the speed control. On straight roadways the steering control works very well, and the car stays directly in the center of the lane for the entire section that it is straight. Curved roadways are not quite as perfect but still work well. Curves of typical angles (90 degree turn, 60 road points, gradual curve) are handled well and the car does drift slightly off center, to the left on a right turn and vice versa, however not a noticeable or dangerous amount. Coming out of these turns the car also functions very smoothly, and by the time the car is coming out of the curve it has returned back to the center of the lane.

Non-typical curves, such as those with angles sharper than 90 degrees do not function as well as standard turns. The car drifts further to the edge of the lane, still staying in the correct lane, but coming close to crossing the line. The car also exits the turn and overcorrects onto the straight away at the end of the curve, which causes an oscillating across the center of the lane until it returns to the center of the lane.

Since the speed control is set such that the car slows down when approaching a turn the car does not drift out of the lane on standard turns. The only time that the car has been seen to drift out of the lane are on hairpin turns, where it is expected the car would either have to have a sharper turning angle, or a slower speed to stay in the lane. However when these things were implemented they negatively affected performance on normal turns, so these non-typical turns were not implemented.

Finally, intersection turns are executed successfully but not well. This is mainly seen when the car is exiting the turn/intersection. The oscillating behavior that is seen when the car exits a sharp curve is magnified when the car exits an intersection, almost to the point where it drifts into the other lane, or off the road. This oscillation also takes some time to go away, but the car does return to the center of the lane after a short time.

Performance Analysis:

The following information expands upon the work done in the project report and quantitatively shows the results of the autonomous driving platform. This includes the methods used in creation of the thesis, as well as the results of the performance analysis.

Methods:

In order to evaluate the performance of the control algorithms implemented in the project it was necessary to add some form of data logging to the platform. This was done through CSV logging through the java interface. Every time that the control loop completes, which happens thirty times every second, a new row would be inserted into a CSV file logging real time statistics about the vehicle. The statistics logged to CSV were:

- Speed limit
- Desired speed
- Actual speed
- Accelerator pedal angle
- Brake pedal angle
- Lane position offset
- Steering wheel angle
- X location
- Y location
- Z location
- Number of lane points on the current roadway tile

Multiple trials were run collecting the above data during each trial, however each trial something about the control platform was altered, such as the control strength of the brake pedal versus accelerator pedal, or feedforward control of steering versus feedback control of steering. This data was then used to build plots to compare and contrast the effects and benefits of different control efforts. The plots were developed using the R programming language and the RStudio software suite.

All of the trials run, logged, and plotted were the result of one lap of a closed loop test track (figure 12).

Results:

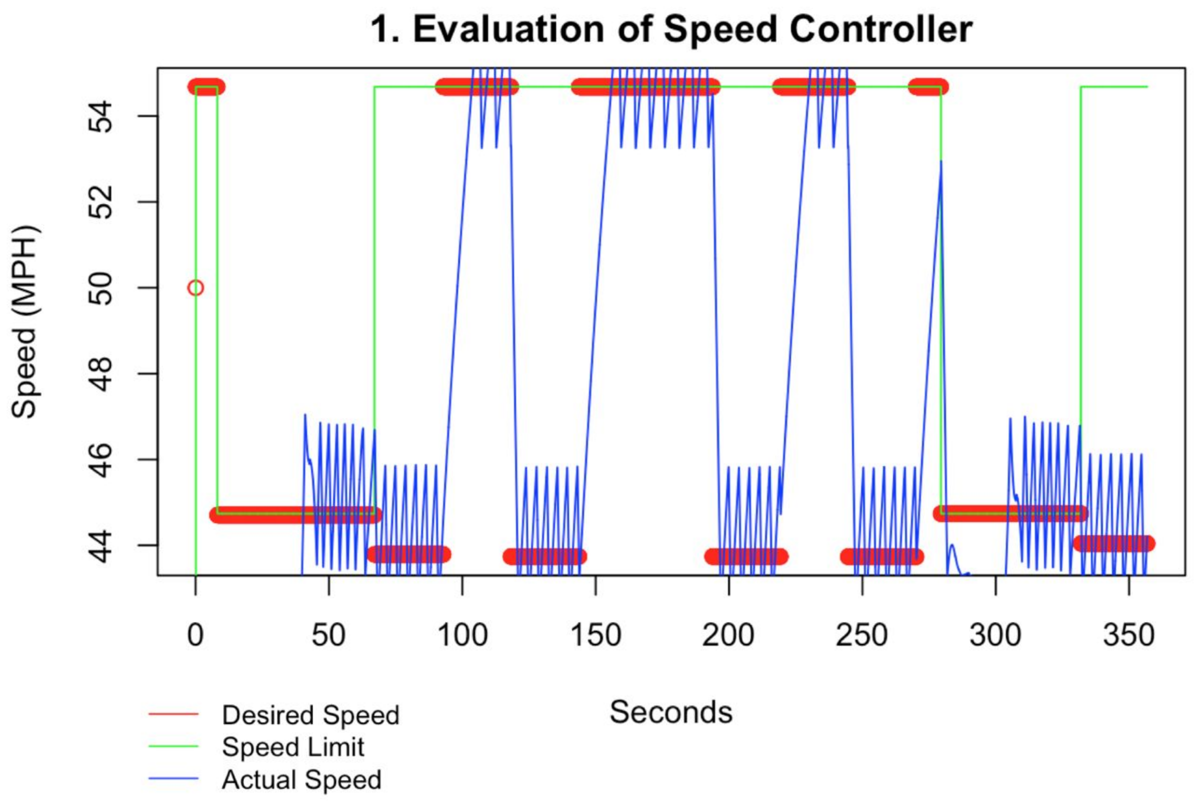


Figure 8a: Plot 1 - Evaluation of the speed controller and visual comparison of desired speed, actual speed and speed limit with final control parameters.

Plot 1 - Evaluation of Speed Controller:

Plot 1 shows a graphical evaluation of the speed controller. This plot sheds light on a couple different aspects of the speed controller. The first of these is the obvious oscillations that exist in the speed, which is easily seen in the variation of the blue line on the plot. This oscillation is in large part due to the way in which the speed controller was designed, in that it has a three mile per hour margin of error around the desired speed. This decision was made because it was more consistent with how an actual human would drive, which would have made more sense had the project team had time to add in ambient traffic.

Also shown in this figure is the degree to which the car slows down around a curve. In the controller design, if there is a curve coming up or on the current roadway tile the car will slow down appropriately such that it will execute the turn correctly. In this plot that speed difference can be seen anywhere that the red desired speed line deviates from the green speed limit line. This shows that the controller slows down the car from the speed limit about 10 miles per hour when encountering one of the right turns on this track.

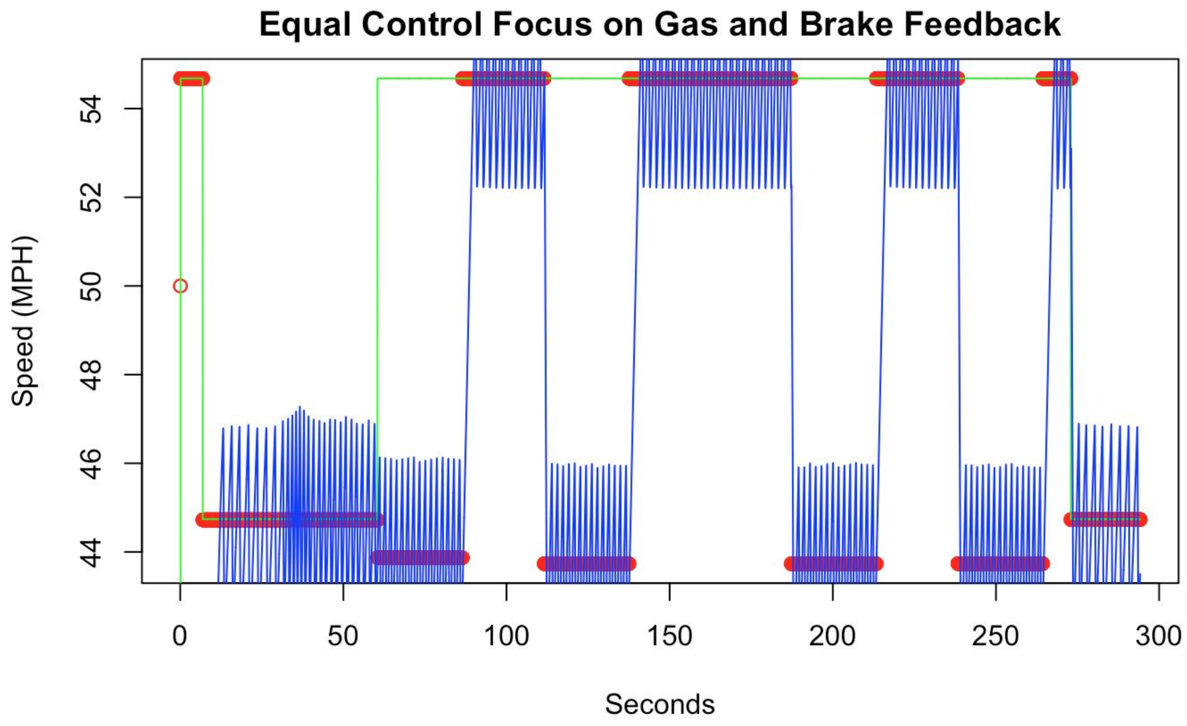


Figure 8b: Plot 1 - Evaluation of the speed controller with altered parameters, where the control strength of the gas and brake controllers is identical.

In comparing figures 8a and 8b it can be shown what a different control effort on behalf of the accelerator and brake controllers does to the overall speed. In figure 8a the control effort of the accelerator controller is 15% and the brake controller is 13%. The rest of the control effort simply comes from the algorithm determining the desired speed of the vehicle. These are the final values used in the final version of the project. In figure 8b the control effort for both controllers is the same, set to 50%. It can be seen from the plot that changing these control values greatly increases the oscillations in the car's speed, and the frequency of those oscillations. This has an overall result in the speed of the car which can be seen from the plot as well as the average speed. The average speed in figure 8a is 45.47, and the average speed in figure 8b is 46.67. This small change in speed wouldn't be expected to have a large effect on the outcome of the car, however the car completes the loop faster in 8b than in 8a by about 12 seconds.

The final interesting aspect of these two plots is how they illustrate the reaction of the car to a change in speed limit or desired speed. Every time that the desired speed changes the car gradually changes speed, shown by the sloped blue lines in the plots. This again is the same as how a real driver would react, and was implemented through both coasting and some use of the brake, which is shown in plot 2.

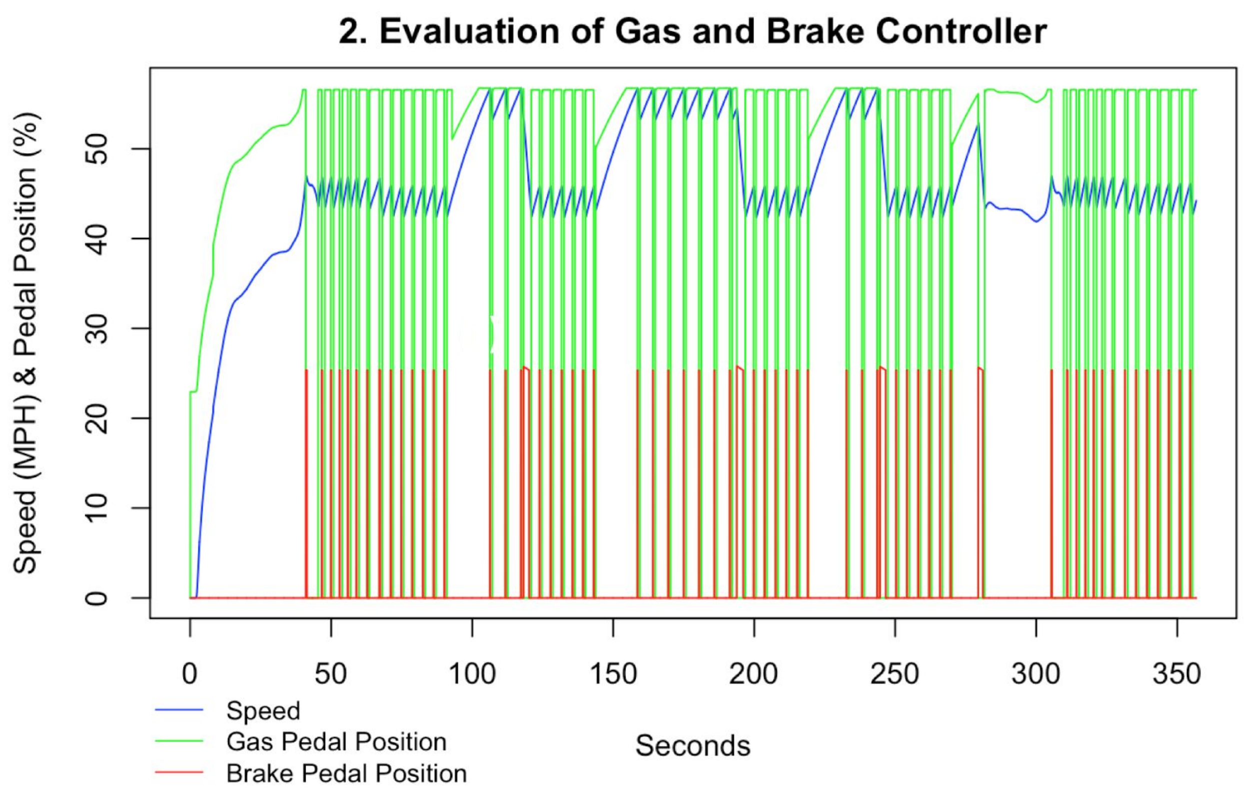


Figure 9a: Plot 2 - Evaluation of the gas and brake pedal controllers with final control parameter values

Plot 2 - Evaluation of Gas and Brake Controller:

Plot two shows the control effort of the gas and brake controllers, and the effect that has on the overall speed of the vehicle. The red and green lines show the amount that each the accelerator and brake pedal respectively are being applied (in percent). The blue line shows the actual speed of the car.

The first interesting aspect of this plot is the first fifty seconds of the data. The accelerator use gradually increases from none to 55% of its total power. As this happens the speed can be shown gradually increasing as well. What is interesting after that however is that the speed reaches the speed limit, and then the upper bound of the three mile per hour margin of error on the speed limit and at that point the accelerator pedal use goes to none and the brake pedal is applied, although always to a lesser extent than the accelerator. The brake is applied just for a short time and then the physics of the simulated vehicle is used to slow the vehicle down through coasting. Then it can be seen in the plot that once the car hits the bottom of the three mile per hour margin of error the accelerator is applied again to bring the speed back up before the process repeats.

This plot shows room for improvement in the overall algorithm for the control of these two components of the car because it is inefficient to have almost a constant back and forth alternation between the accelerator and the brake pedals. The behavior should be more gradual overall similar to how it is in the first fifty seconds and around the three hundred second mark when the car goes back into the hilly curved section of the map.

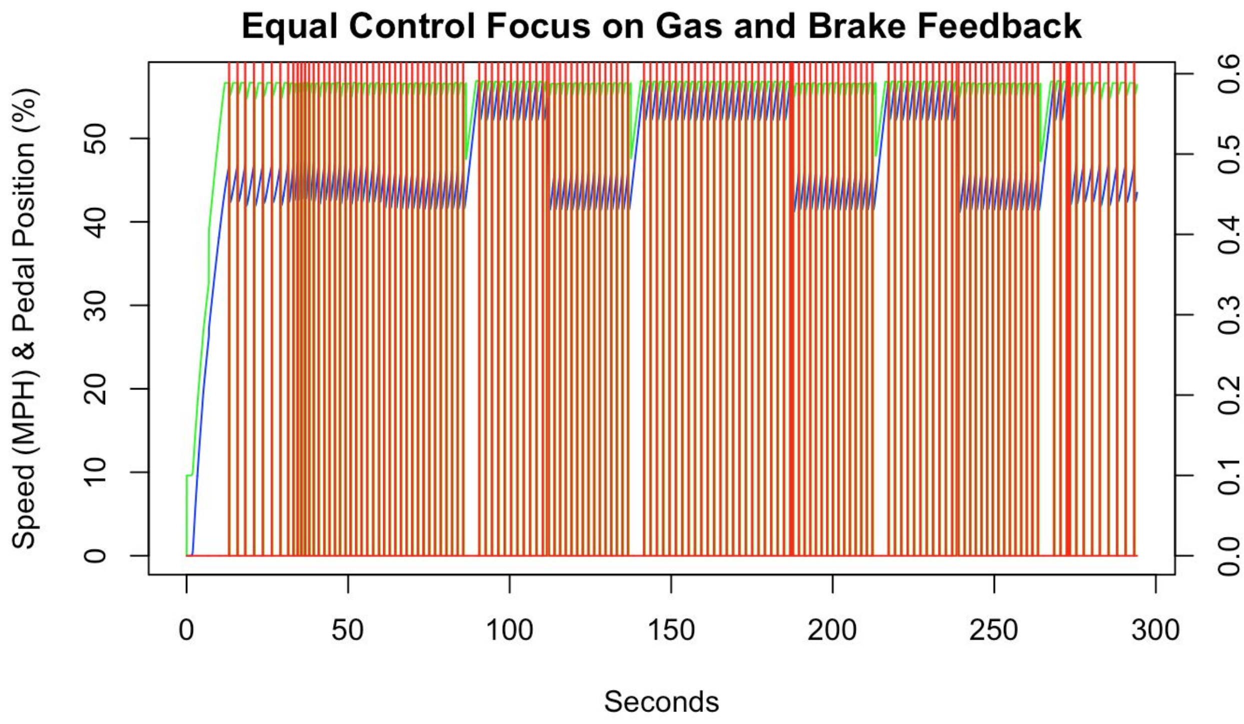


Figure 9b: Plot 2 - Evaluation of the gas and brake pedal controllers with equal control parameter values

Figure 9a shows the results collected from the final implementation of the platform, while figure 9b shows the results collected when the control strength of the brake and gas pedals is set to be equal, as in figure 8b. The first contrast between these two plots is again the frequency change in oscillations that was also seen in the data from plot 1 when figures 8a and 8b were compared. This is a result from the increased control strength in this second plot as the accelerator and brake pedals are being applied more heavily so the speed changes are faster. The next noticeable change in this second plot is the fact that the brake pedal has much more focus in this second plot. In figure 8a the brake pedal was never applied further than 25%, however in figure 8b the brake pedal is always applied at 60%, due to the higher control strength. The gradual increase in acceleration that was seen across the first fifty seconds in the original plot is greatly decreased in figure 8b due to the increased control effort of the accelerator in this plot as well.

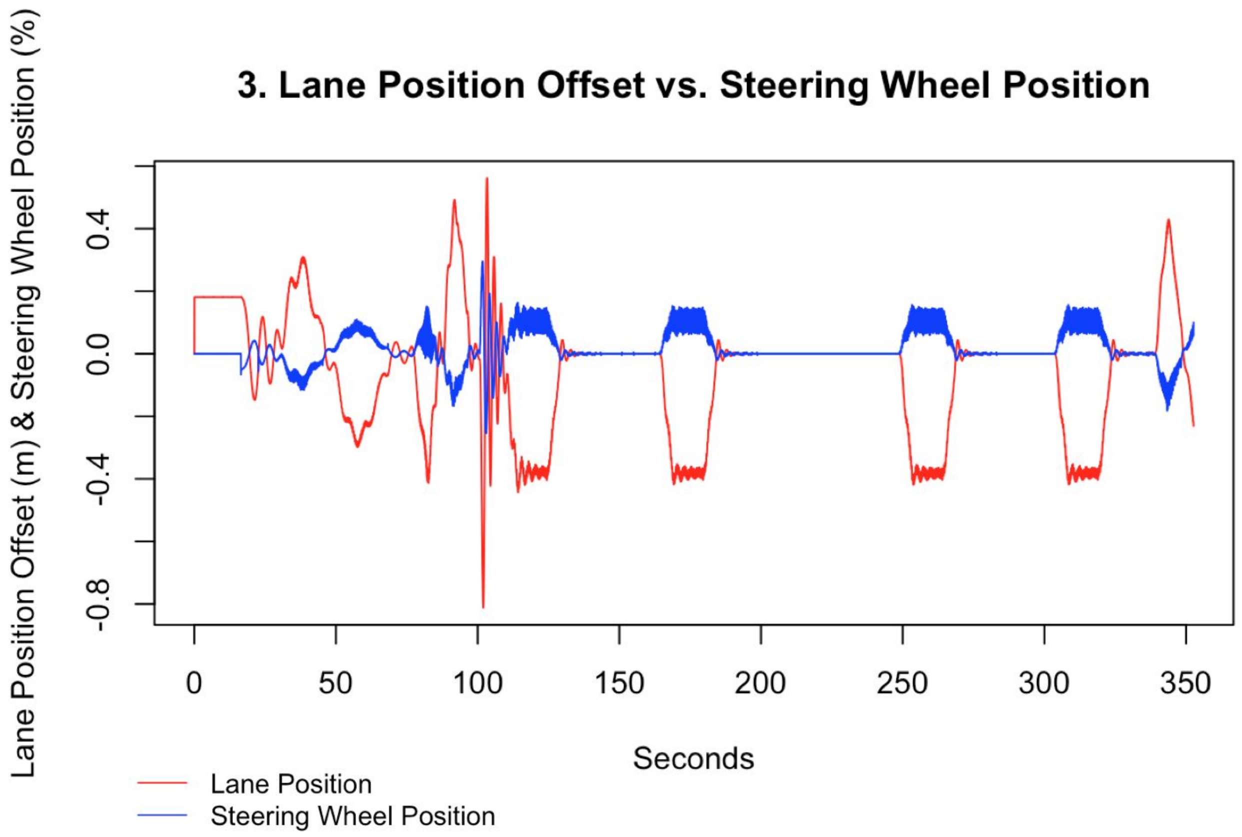


Figure 10a: Plot 3 - Visual comparison of the lane position offset and the steering wheel angle with final control values

Plot 3 - Lane Position Offset vs. Steering Wheel Position:

Moving on from analysis of of the speed, accelerator, and brake controllers leaves the steering control. Plot three shows something really interesting about the steering controller. This is the way that it is possible to see in the plot the controller fighting against the car's deviation from the center of the lane. The red line in the plot shows the car's deviation from the center of the lane that it is currently in (in meters), while the blue line shows the steering wheel's position as a percentage of its total turning capability.

What's interesting about this plot is how it quantifies the oscillations that can be seen in the driving simulator as it runs. For instance, there are slight oscillations in the car's lateral location in the lane when it goes around a curve (e.g. 100-150 seconds, 150-200 seconds), that can be seen in this plot. It can also be seen that while it appears that, from watching the simulator, the car stays fairly centered around curves, this is not completely true because the plot shows a slight lane offset of about 0.4 meters around the curved parts of the road.

Another fact that can be inferred from the oscillations is that while the oscillations do exist around turns the amplitude of the oscillations decreases as the turn is executed. This can be seen in the red line, or the lane position offset line. For instance in the turn that happens at about 125 seconds it is easily seen that the car is settling out around the turn as the car moves through the turn. Also what can be seen in this line is that as the car moves out of the turn it returns back to the center of the lane almost immediately, back to an offset of 0 meters.

Something else to be said about this plot is that while it may appear that the car sometimes moves very far out of the lane, it is important to note that the lanes the car was on are 3.6 meters wide in total. Combine this with the fact that the car is a width of 1.7 meters, and then take the largest lane displacement of 0.8 meters from the center, that still puts the car 0.2 meters away from the side of the lane, which comes out to a little less than 10 inches from the side of the lane. While this is far from perfect, it is still within the margin of error that would be acceptable for driving practices.

Multiple different control values were tested to see which had the best performance on steering control and these different control parameters mainly affected the dependance on the feed forward control versus the feedback control. Feed forward control looked at the road ahead of the car and the shape of the roadway and adjusted the steering as a result of that. The feedback control looked at previous steering angles and dampened large changes in steering wheel positions so as to not jerk the steering wheel back and forth.

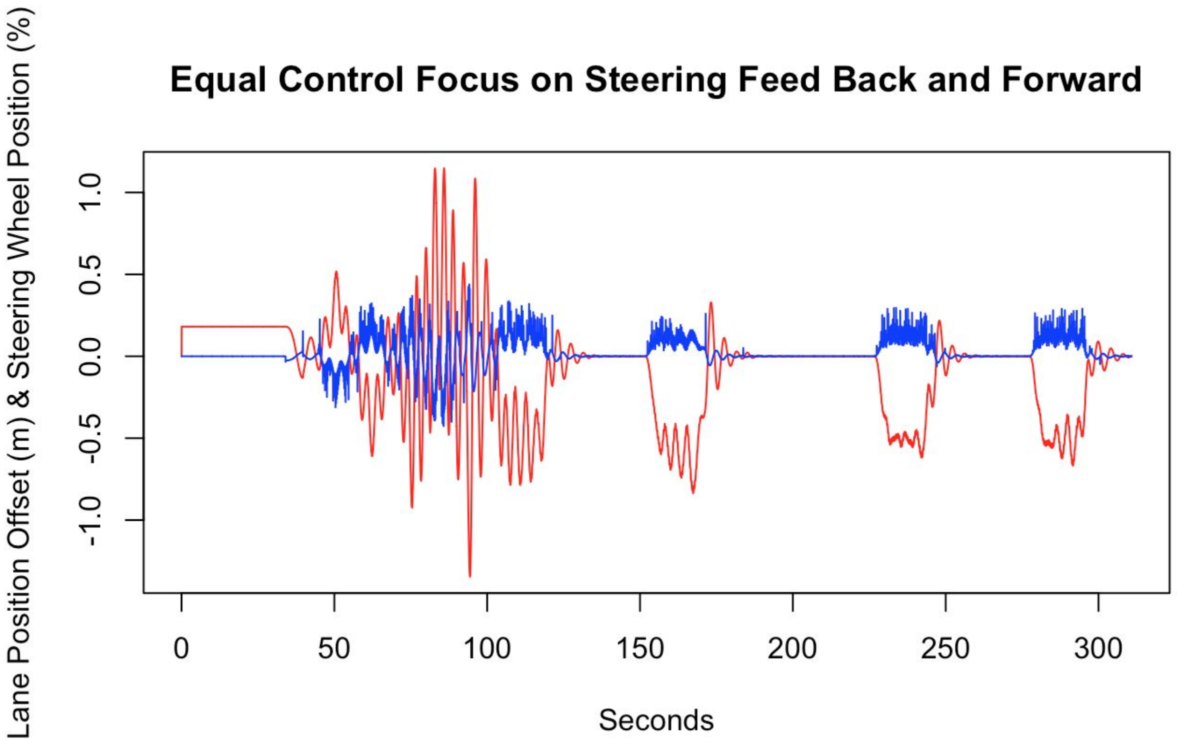


Figure 10b: Plot 3 - Visual comparison of the lane position offset and the steering wheel angle with equal focus on feedback and feedforward control

Figure 10b shows the results of having equal control weight on the feedforward versus feedback control. This plot shows the drawbacks of a control strategy based off of these control weights as the magnitude of the oscillations are much larger and more erratic. On top of that the overall scale of the graph showing the lane deviations is larger. On this version the largest lane offset is 1.5 meters from the center of the lane which puts the car well into the other side of the lane or off the road, depending on which side the deviation is on. On top of that the car does not return to the center of the road as well as the other version with the final control parameters for feedforward and feedback. This poor return to the center of the lane can be seen around 100 seconds on the plot. The larger oscillation about zero lane position offset does not exist in the other control version, making this one inferior to the final control platform.

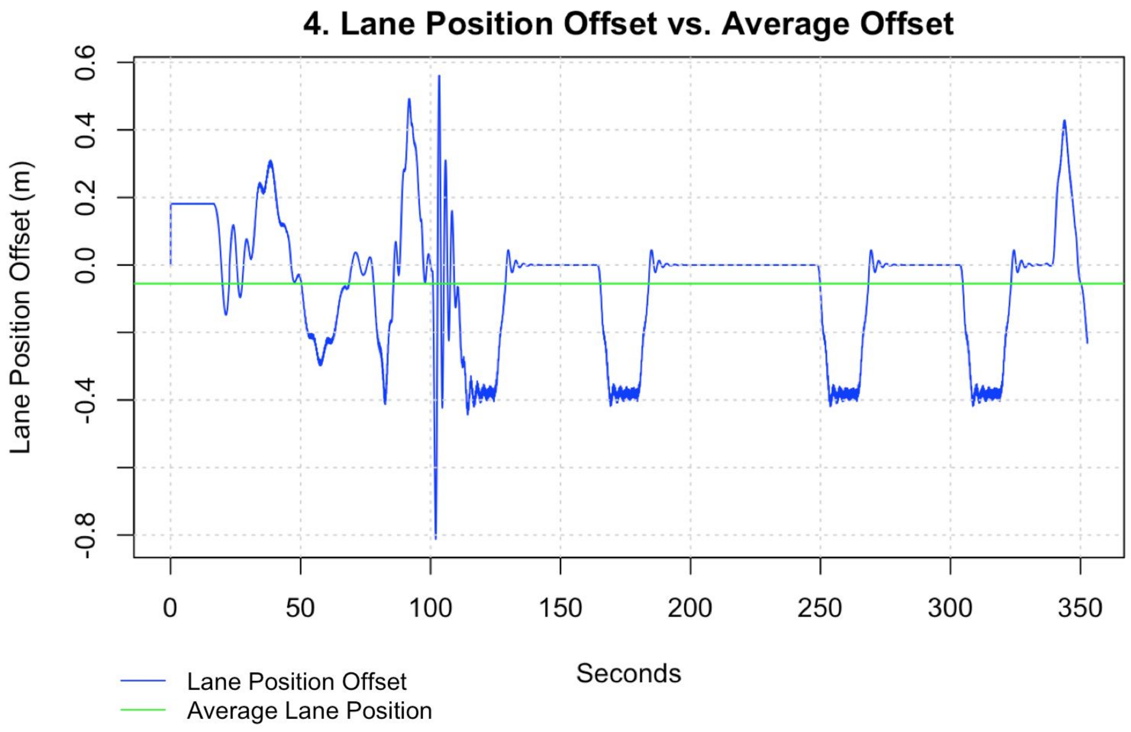


Figure 11a: Plot 4 - Visual comparison of the lane position offset and the average lane position offset with final control parameter values

Plot 4 - Lane Position Offset vs. Average Offset:

This final plot is an extension upon plot 3, showing further the impact of the oscillations on the autonomous driving platforms overall performance. This plot shows the same lane position offset as in plot three but then compares it to the overall average lane position offset (the light green line). What is interesting about this plot and the two figures representing the plot (figures 11a and 11b) is that even though there are large deviations from the center of the lane at times, the overall average lane deviation is not very large. This in part shows that the controller is doing an acceptable job, but also reflects upon the design of the map, where there is a good portion of straight road on the track.

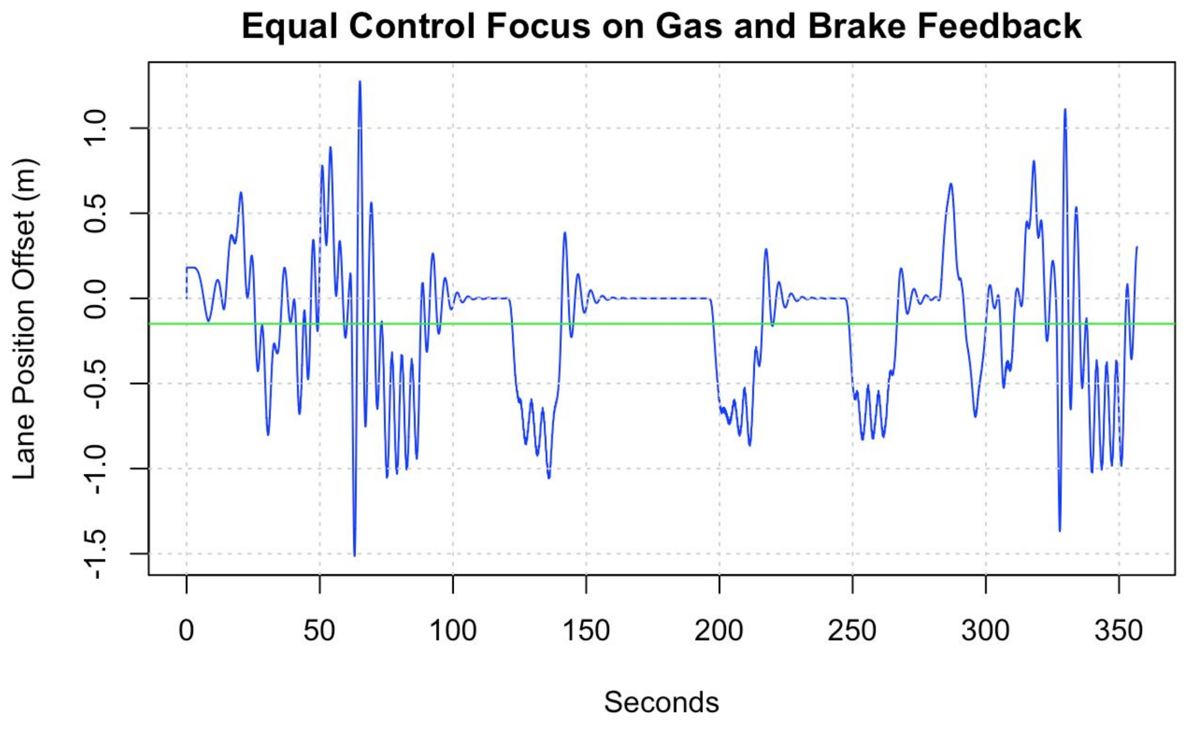


Figure 11a: Plot 4 - Visual comparison of the lane position offset and the average lane position offset with equal control parameter values

Comparing the two figures, 11a and 11b again show the superiority of the final control platform (11a) when compared to the prototype (11b). The average lane deviation in figure 11a is -0.05m compared to 00.15 meters in 11b, a difference of 0.1 meters. This is not a particularly significant distance however it is notable, and does show the better performance of the final control platform. These two plots also make it easier to see the oscillations in lane position compared to the plot three since it is not overlayed with the also oscillating steering wheel position. Now when comparing the two figures (11a and 11b) it is easy to see that the oscillations are larger and more frequent on the prototype control platform than they are on the final version.

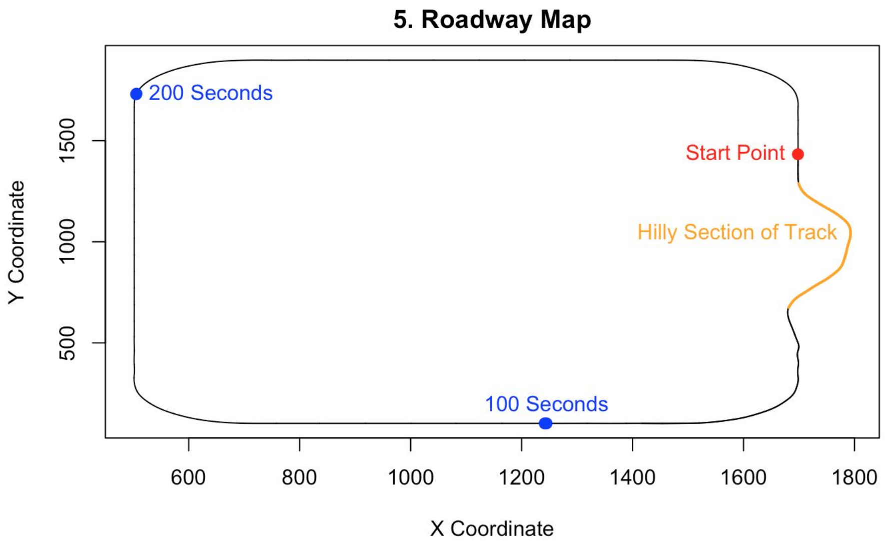


Figure 12: Roadway map showing the startpoint, checkpoints, and hills in order to compare to other figures

Plot 5 - Roadway Map:

This final plot is really for a reference when looking at the other plots. This plot was built using the x and y coordinates of the vehicle during a test run. Markers have been added in to show the start location, certain timing checkpoints as well as components of the map. The car travels in a clockwise direction around the map from the start point. The first and only feature other than turns is a section of the map that contains a few small bends as well as some hills. This can be seen in figure 13 which is a screenshot of the map building software. The tile on the right side of the map is the one that contains the small bends as well as the hills.

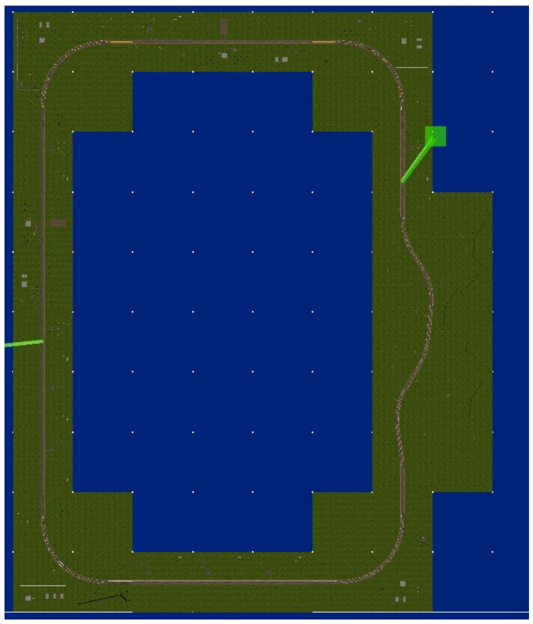


Figure 13: Roadway map showing two start points, screenshotted from the hyperdrive map building suite.

Performance Analysis Conclusions:

Through analysis of the above figures and the discussion that accompanies them it is clear to see that the final version of the control platform is quantitatively better than the earlier version with balanced control strengths when it comes to both speed and steering control. From looking at plots one and two, the evaluation of the speed controller and the evaluation of the gas and brake controllers respectively, it can be seen that the performance of the platform strongly reflects the decision made during development of the controllers and shows a good performance overall. In plot one, the evaluation of the speed controller, the fluctuations in speed show the imperfections that happen when a human driver is behind the wheel, and also stops the car from having very juddering performance. This juddering is shown in the earlier version of the platform where the speed fluctuates much more frequently, which gives an unfavorable riding experience. Along with lack of juddering the acceleration/deceleration performance is also better in the final version, shown by more gradual speed increases/decreases and a gentler slope as the correct speed is approached, which is how a user would expect performance to be.

The same results are shown in the accelerator and brake controllers, as what is really important here is that they have a positive effect on the overall speed. They could be somewhat improved to have a wider range of pedal position (amount that the pedal is applied) however even without that improvement the resulting speed is acceptable. The same gradual increase in speed as the accelerator position varies is shown in plot two as in plot one and again the performance is much better in the final version than the earlier version of the platform. The same juddering that was seen in the earlier version of the platform in plot one can again be seen here, and the cause, being the frequent alternation between the accelerator and brake pedals, makes it apparent why the speed fluctuates so frequently. Overall however the speed controller functions fairly well, and from the information shown by the plots, the most valuable improvements to be made lie in the steering controller.

The largest room for improvement when it comes to the control platform falls under the category of the steering control, which is easy to see from the figures and data above. Plots three and four represent the analysis of the steering control and show some of its shortcomings, despite the fact that, like the speed controller, it is an improvement from earlier designs. In plot three the biggest drawback of the steering controller is shown in the oscillations that occur whenever the car goes around a turn or as it passes through the hilly/curved section of the track. The oscillations in both of these types of sections are smaller than those in the earlier version of the platform, shown in figure 10b, however they could still be better. The largest problem shown by figure 10a is the poor performance through the curved/hilly section of the track. This is the only section where the oscillations carry the car from almost going out of the lane on the left to almost going out of the lane on the right before settling out. This could be improved through a more sophisticated control platform. Despite this issue, the final version shows decreased oscillations from the prior version in this curved/hilly section as well as around turns in general. On these sections there is an overall offset from the center of the lane, however it is fairly negligible when the size of the lane and the size of the car is taken into account.

The main purpose of plot four was to demonstrate that when quantitatively evaluating a control platform in this way it is important to look at all of the data instead of just averages. In both versions of the control platform the average lane deviation is relatively small which would reflect strong performance of the platform. However this is not true when the lane deviation is plotted across the runtime of the simulator. So while averages are easy metrics to compare in this situation it is not as applicable. Overall the figures and data show that while there is still work to be done on the control platform, the improvements of the final version over earlier version of the platform are strong, and provide a strong foundation for future improvements.

Conclusions:

The following conclusions are adapted from the original project report for use in this thesis to provide the reader with information about how future improvements could be made to the platform as well as a general conclusion of the work done during the capstone project.

Future Work:

Algorithm Improvements:

The current control platform implementation is fairly basic, and was based off of trial and error to see what changes to the control algorithm would result in the best qualitative results. By qualitative results, what the project team looked at was the output of the simulator on the three displays, and how well it looked like the car was driving. There is a lot of room for improvement when it comes to the control algorithm, mainly including some more advanced controls functions.

One such control function that could have greatly increased the successfulness of the driving simulator would have been the inclusion of the transfer function in the steering control. Using the transfer function could have greatly increased steering stability, especially when the car is coming out of a turn or off of an intersection. Currently the subject vehicle oscillates across the lane for a few control loops until it evens out at the middle of the lane, however it is predicted that using the transfer function could greatly cut down on these oscillations and swerves across the lane.

This oscillation problem also relates to the last improvement that the project team could see making, which has to do with the way that the control platform handles intersections. As previously stated, the subject vehicle could handle exiting intersections much better than it does, with fewer oscillations through use of the transfer function, but on top of this, the handling of intersections could be much more intelligent in general. Currently when the car enters an intersection with the intent of turning left or right, the steering wheel is set at a fixed angle until the car exits the intersection (passes the first lane point on the new lane). This works fairly well currently but it also assumes that the car stops at exactly the same distance from the intersection every time, and that every turn, left or right, is identical. What would be a better implementation is to handle each intersection differently, based off of the distance from the start of the intersection, the start point, the exit point, and better mapping in the interior of the intersection.

Currently the simulator does not provide any information about the inside of the intersection such as lane position offset, which is why the project team decided to go with the simpler method of a fixed angle. However given more time there would be the possibility to essentially set up a grid of coordinates so that the car would understand where it was in the intersection and would be able to navigate intersections much better.

UI Improvements:

The current state of the UI is somewhat unpolished, and moving forward there is definite room for improvement. First, the aesthetics of the Control UI could be improved to display the information in a more user friendly format. For instance, as the length of the strings containing the numbers being presented changes, the formatting of the UI changes which makes them harder to read.

As far as functional improvements go, something the project team wanted to include but didn't have time for was the disabling and enabling of the buttons on the Navigation UI at intersections. For example, currently the user can click any of the three buttons, left, straight, or right at any time while the simulator is running. This can sometimes cause unpredictable behavior as the buttons are only meant to be pressed when the car is stopped at an intersection. Therefore the team wanted to disable the buttons until the car stopped at an intersection, and then only enable the buttons for the directions that it was possible for the car to go. For instance if it was a three way intersection and the only options for turning were either left or right, the straight button would be disabled. Currently, without this implementation, the user can select an invalid direction which crashes the control code because it tries to retrieve information from the simulator that doesn't exist.

Simulator Improvements

The simulator had a plethora of additional features that would allow for the car to be tested under more life-like conditions. Cars, pedestrians, and even animals could be scripted to do certain things when the autonomous car reaches a point to test how it can handle obstacles and other ambient traffic. In the end there was never a need to script any active traffic or obstacles as the vehicle was not designed to handle such conditions.

The simulator also allowed for variable weather conditions which could cause ice like traction on the roads or more difficult driving conditions. Since the simulator is designed specifically for training drivers, there are thousands of combinations of environmental variables that could be added to further test the car under all possible roadway conditions. The style of road used most frequently in testing were rural. This consistency made it simpler to test under ideal conditions. More urban and suburban environments would be an effective way to test the autonomous vehicle under tighter and less free range conditions, but for our design the curvy and ideal rural roadways were the best option.

Evaluation of Progress:

By the end of the year the project team accomplished a lot of what they wanted to accomplish, adding a good amount of functionality to the platform. In general, however there were definitely things that had been predicted to be completed that were not. Successfully implemented was the speed control as well as the steering control on closed loop tracks of any shape or size. This took longer than the team originally thought that it would simply because any small change to one control strategy required going back and changing the other. This back and forth took longer than expected and these two control strategies ended up taking almost all of the first half of the year.

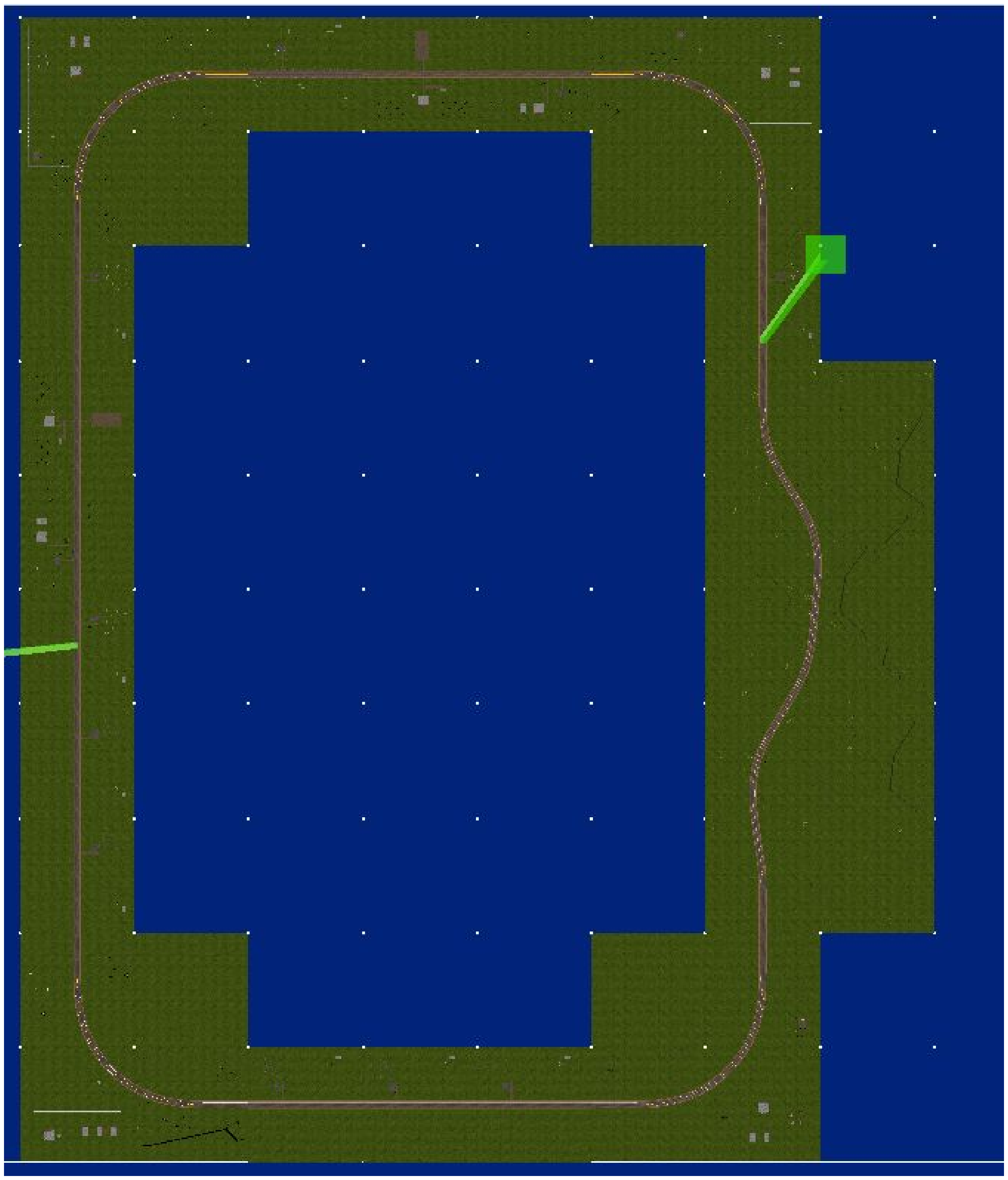
Navigation control, minus some small improvements previously mentioned, was also completed and allows users to select which direction they would like to go at any given intersection. The control UI is also completed, successfully displaying live statistics in the UI and logging them to a CSV for interpretation.

Project goals that are still in progress would include intersection handling. Many improvements could be made to the way that intersections are handled, and with that fine tuning improvements could be made to the steering control for when the car leaves an intersection. However the simulator does successfully navigate almost all intersections after stopping at them, which was something that had not been implemented at all before the project team started work, which is why this goal is seen as partially complete.

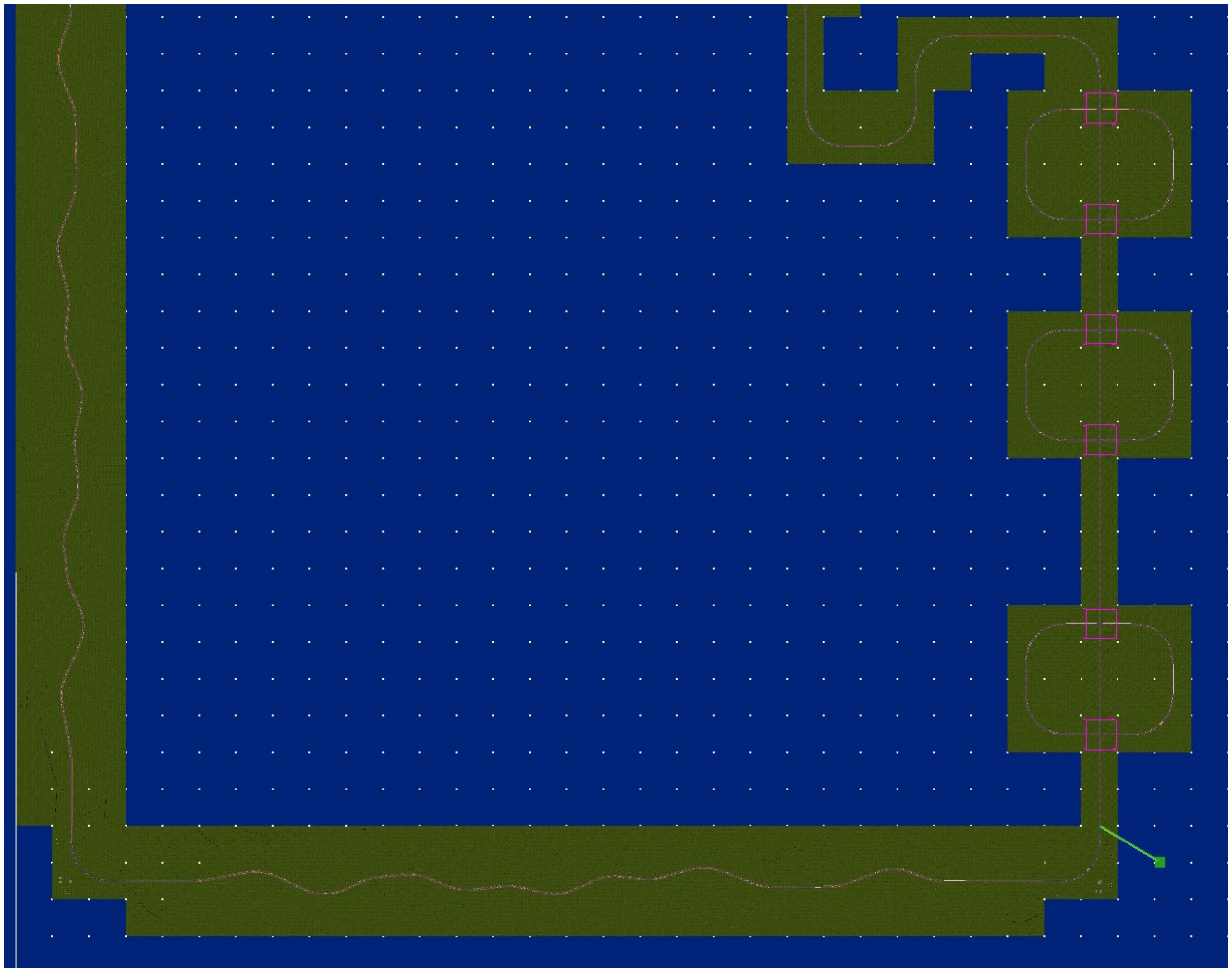
The implementation of stopping the car at an intersection as well as executing a turn at an intersection took much more time than originally expected mainly because of the limitations from the simulator about intersection information available. Because of this, the rudimentary intersection handling that is currently implemented took a great majority of the second half of the year, as well as preparing for the Undergraduate Research Conference. Due to these setbacks, there were a couple tasks that the team hoped to get to that had originally been goals at the beginning of the year. These mainly included adding more real world situations such as other traffic, stopping if pedestrians were crossing the road, or other boundary circumstances such as this.

Even without those extra implementations however the project team still hit a few of the largest milestones such as steering and speed control, as well as stopping at intersections. These provide a strong foundation for future groups to build upon to improve the project even further and to hopefully end up with a completely self sufficient autonomous vehicle control platform.

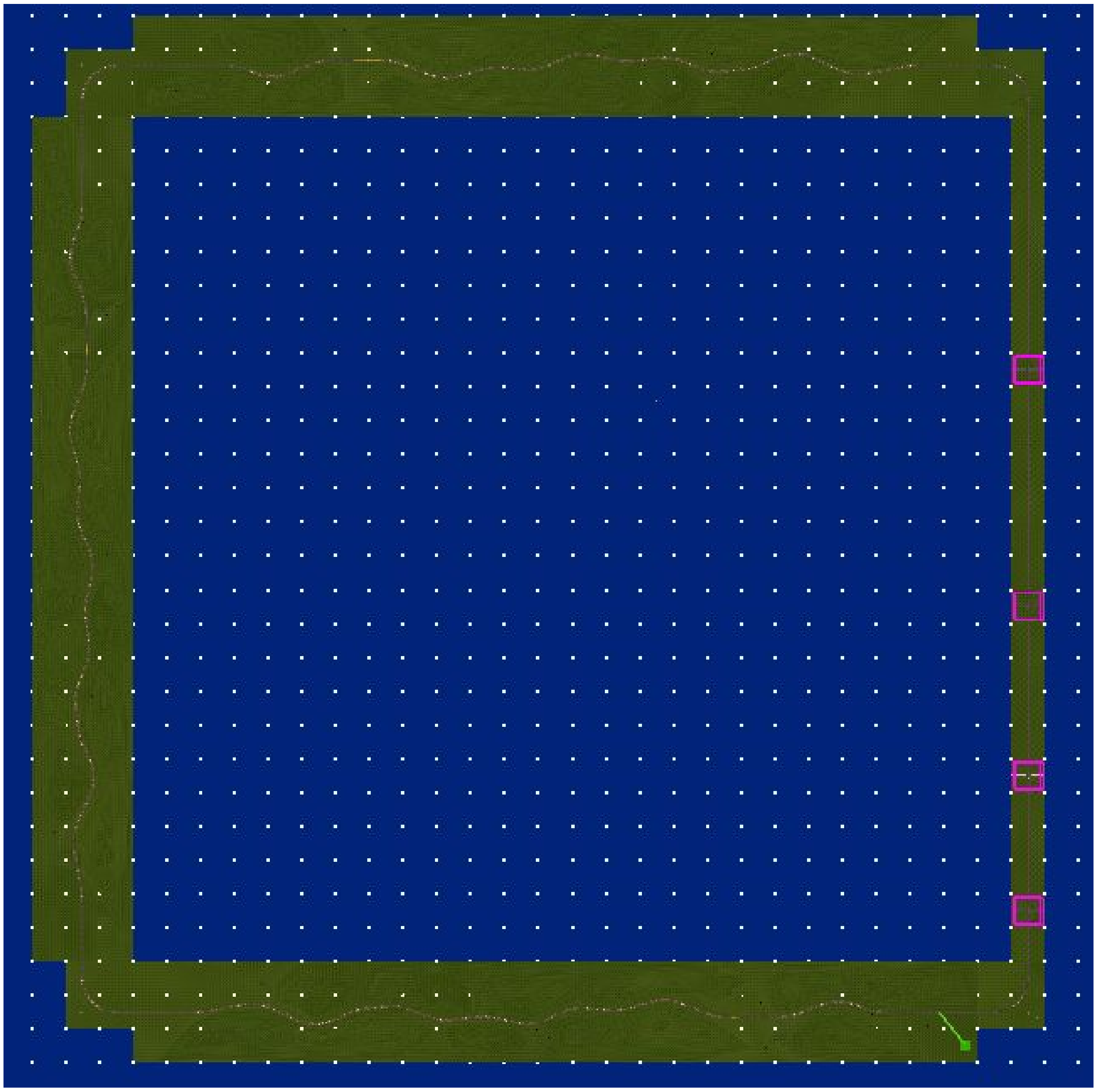
Appendix 1: URC and Test Maps



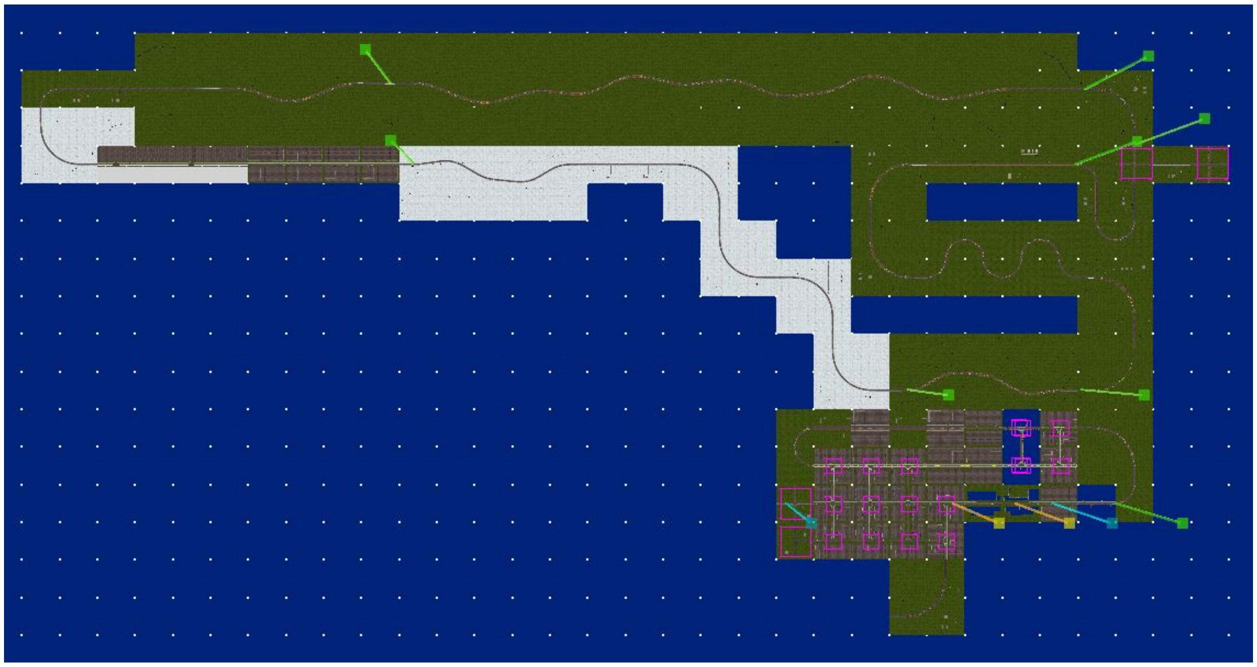
1. Simplest URC test track. This is a short closed loop track designed to completely minimize any problems and randomness. It features a familiar curved road, straight roads and 90 degree corners



2. URC Large Intersection. This is a track featuring two very long (3 mile) backroads and a straight section with a series of intersections that lead back to the primary track. This test allows for the car to go along the backroads for a very long time, or go through a series of intersections that loop back to the primary closed loop to prove the car's capacity to take lefts, rights, and straights at stop signs.



3. Similar design to the URC large intersection, this one includes 3 of the 3 mile back road tracks as well as several intersections at different distances to test stopping distances upon approach.



4. AutonomousTestMac: This map was not for URC purposes and is the most elaborate test map. It features large back roads to test the cornering algorithms as well as very tight 90 degree turns to push the limits of the turning algorithms as well as speed control. The busy section in the bottom is an urban section of the map with a grid style layout, similar to what one would expect in a modern city. This map is where we most heavily tested and practiced with scripting other cars and pedestrians to take actions depending on where the autonomous car is. Routes were designed that ambient traffic could follow and foot traffic was added.

Appendix 2: Java Control Software

Running the Autonomous Control:

The autonomous control code lives on the java server computer in the Project54 lab in Morse Hall on the UNH campus. The most recent version of the control code can be found at the following path on the machine: C:\Users\Project54\IdeaProjects\UNHAutoPilot\UNHAutoPilot-UI\AutonomousCar. To open the code open the file named AutonomousCar.iml. This is an IntelliJ specific file that will launch the IntelliJ java IDE which is the development IDE that the project team used. To run the code, click the green play arrow at the top right of the window.

When this happens the IntelliJ console will print out a message saying that the server has been started, and the two UI windows, the ControlUI and NavigationUI will open. Once this has happened move over to the Simulator Control computer. Launch the Dashboard app on the desktop, select any of the maps that are listed in the previous appendix and hit start. Once the simulator has loaded on the three simulator screens, the IntelliJ console on the java control computer should say that the connection has been established and this is how the user knows that the control computer is ready to drive the car autonomously. To start the car in the simulator push the red lowest left button on the steering wheel and the car will begin to drive autonomously.

Java Code Organization:

The Java code organization is broken down by the category it relates to. Anything that pertains to the actual autonomous control of the car (steering, speed, etc.) is located in the AutonomousControl.java file which can be found in the AutonomousCar folder of the source files. If something needs to be queried from the simulator, or the control code is pushing information to the simulator, those files are located under the folder “SimConnection”. All of the java classes in this section are prefixed with “SIM….java” so that the user knows that these are for java-simulator communication. These classes are things that directly get and or give information from simulator components such as the car’s current speed, or the car’s brake pedal angle. UI code for the navigation UI and feedback/feedforward UI is can be found under the “UI” folder. There are separate classes for both the navigation UI and the control UI, NavigationUI.java and ControlUI.java respectively Finally, the SimUtilities folder contains files that pertain to classes and methods that are useful globally in the project, such as the Location.java and VehicleInfo.java classes.

The entirety of the control code is encapsulated within the AutonomousControl.java class, so a more in depth description of the code in this class is necessary. Since the control code happens in a control loop that gets called externally it is necessary to use a large number of global variables which are the first thing that are encountered in this file. The next important portion of this class is the getFeedback method. This method contains the code to query information from the simulator for use in the control loop. There is also a section of this code that only gets certain information once per road tile, such as the number road points on the tile, or whether or not the next tile is an intersection.

The largest method in this file is the doControl method. This method acts as a large control loop that runs thirty times every second. This method handles everything from waiting for user input at intersections to adjusting the speed via the brake and gas or managing the steering wheel angle based on current and past information about the car. The actual control code for the speed and steering are handled in two separate methods, doSteering and doSpeed which both get called from the control loop. There are many other important methods in this class however they are self explanatory when looked at in the context of the code, and with the given information above it would be able for another project group to pick up the project where this year’s team finished.