Cooperative vs Non-Cooperative Autonomous Driving

Authors: Tito Anammah, Serris Lew, Kylee Santos
Electrical and Computer Engineering, Carnegie Mellon University

Likely to be the future framework of transportation, autonomous driving has been the forefront of much research in the artificial intelligence field. We look to explore potential improvements to the current system, by experimenting the effect of vehicle-to-vehicle communication on the road. Given the scope of our project, we simulate with different scenarios of cars on a figure-8 track. Different information constraints are given to compute decisions for each vehicle. We measure and compare the throughput between the non-cooperative and cooperative approaches in order to demonstrate clear improvements to the system.

Index Terms — *Cooperative mode*: cars communicating with one another to make a collective decision to optimize the overall system goals. *Non-cooperative mode*: cars making individual decisions based off of immediate surroundings to optimize its individual goals. *Throughput*: Number of vehicles that pass through central lane given a specific amount of time.

1 INTRODUCTION

Autonomous driving will likely revolutionize the transportation industry in the next couple of years. Even more pioneering is the current research on cooperative autonomous driving. We aim to explore the benefits of vehicle-to-vehicle (V2V) communication and the advantages of cooperative decision making between autonomous vehicles. Having multiple vehicles cooperate could not only improve safety but also decrease traffic congestion.

There has been a lot of research on autonomous vehicles that make decisions solely based on their immediate sensor input. Simulating V2V communication implicitly extends a vehicle's sensory range and allows it to make more informed decisions based on its environment. It provides additional information such as another vehicle's intended path that cannot be fielded through sensor data. For demo purposes, we will create a simulation consisting of a figure-8 track setup with multiple cars on each loop. The center lane will be shared by cars from both loops, and our goal for this project is to achieve at least a 30% increase in throughput compared to a non-cooperative scenario.

2 DESIGN REQUIREMENTS

Given the size of our simulation window, we scaled the size of the track and vehicle to include around 9 - 13 vehicles at a time. The other parameters were given realistic numbers and converted to pixels with the same scale.

With a scale of 1m to 25 pixels, we decided on a track size of 1000 pixels by 550 pixels with an outer radius of 275 pixels. A breakdown of the vehicle and track sizes are shown on Figure 1.

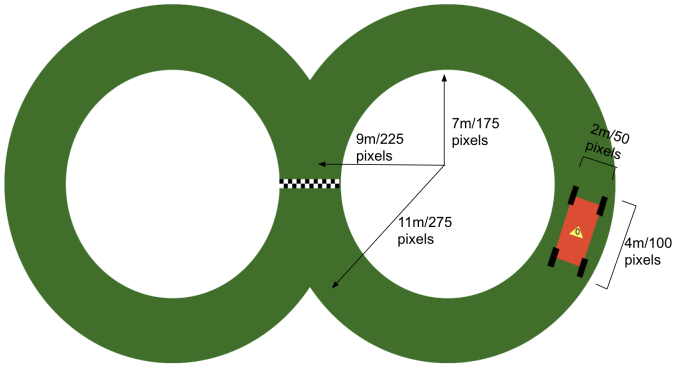


Figure 1: Track Scale

We used these metrics in order to determine a minimum following distance between vehicles so that they would still be able to safely come to a stop. This includes buffer distance, optimal velocity, maximum acceleration and maximum deceleration. Buffer distance is the distance between the stopped vehicle and the obstacle in front of it. Optimal velocity is also the maximum velocity; with no obstacles in front of it, a vehicle will plan to move as fast as possible. The specific values for these design metrics are shown in Table 1 below.

Number of cars	9-13
Optimal velocity	0.2 m/s
Maximum acceleration	0.3 m/s^2
Maximum deceleration	0.2 m/s^2
Buffer distance	7 m
Length of each vehicle	4 m
Width of each vehicle	2 m
Circumference of one track	69.1 m
Outer radius of one track	11 m
Inner radius of one track	7 m
Lane width in each track	4 m

Table 1: Metrics for track design

Another project requirement is to validate the performance of our path planning algorithms. As mentioned

above, our main goal is to achieve a 30% increase in throughput in the cooperative case compared to the non-cooperative case. Along with throughput, we wanted to compare other changes in behavior and decision making. Similar to throughput, we aim to have a 30% increase in average velocity and a 30% decrease in waiting time; waiting time is defined as the total time each vehicle's velocity is 0 m/s. We also aim to have a 15% decrease in acceleration and deceleration; as the vehicles share their intended path to one another, it is expected to reduce sudden, sharp movements. With vehicle to vehicle communication, the cooperative approach should ultimately improve traffic flow.

3 ARCHITECTURE OVERVIEW

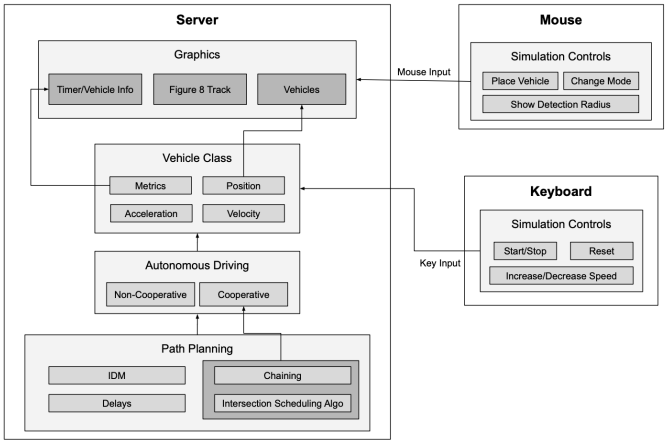


Figure 2: Block Diagram of our system

Our system architecture is broken down into 2 main components: the server and the user input. The block diagram in Figure 2 shows all of the different modules and how they connect and interact with one another.

3.1 User Input

The simulation functions in two modes: the customizable setup mode and the test mode. In the former mode, the user can interact with the simulation interface to fix different settings of the simulation. The user can initialize vehicles to locations of their choosing by clicking on the track, allowing them to try out the simulation with varying car distributions. They can start, stop, and reset the simulation using the keyboard, and also toggle between the cooperative and non-cooperative control modes. The detection radius button also gives the user the option to view how far each vehicle detects its surroundings.

3.2 Graphics

The server provides graphics as a visual representation of the simulation. In addition to the quantitative metrics, we wanted to visualize the difference in movement between approaches. For demo purposes, we included a checkered intersection line that constituted as the throughput marker, a timer to display how long the vehicles are moving and a total loop counter which is the throughput. Figure 3 shows an example scenario with multiple cars moving on the track. Each car is labeled with an id that can be matched on the information list on the top left corner of the screen; each vehicle's velocity is included. As the vehicles make a collective decision together, the cooperative case also includes the number of vehicles that are behind it.

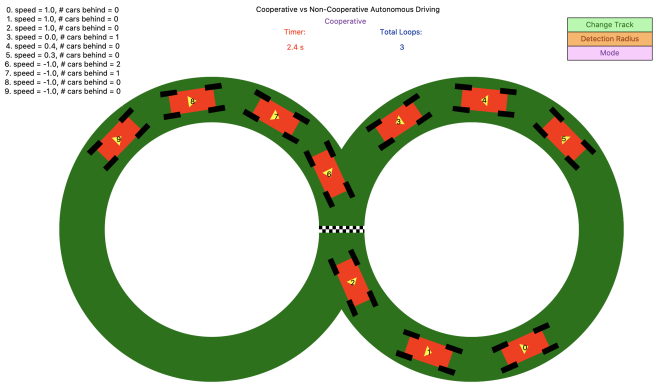


Figure 3: Graphic representation of simulation

3.3 Vehicle

The Vehicle class allows the simulation to provide multiple vehicles that act individually. This class houses information related to each vehicle such as its current location, speed, and acceleration. Beyond the basic information regarding the vehicle state, this class also keeps track of communicated data such as the chain to which a vehicle belongs and its corresponding lead vehicle. It also keeps track of the vehicle's current estimation of the number of vehicles behind it (when the vehicle approaches the intersection). This sort of data gets updated as vehicles communicate with one another. Lastly, we have each vehicle keeping track of certain metrics such as the number of times it has passed the intersection. This information is aggregated across all the vehicles at the end of a simulation.

3.4 Path Planning

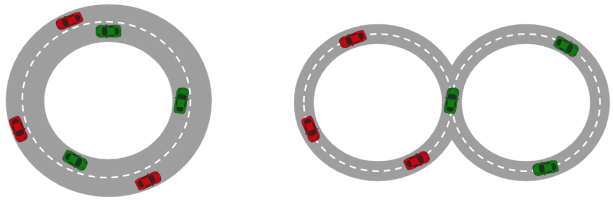
The path planning component controls each vehicle's movement at every time-step. At the base level, each car is controlled with the Intelligent Driver Model (IDM) which is a time-continuous car-following model; this is used for both the cooperative and non-cooperative case. For the cooperative case, we incorporate other factors into the control mechanism such as car chaining and the intersection scheduling. For example, the intersection scheduler sends signals to cars indicating whether or not they can pass the intersection and these signals influence a car's movement

accordingly. This component has a full view of the system but applies information constraints on all the vehicles so each vehicle has only a limited view of the system i.e the vehicle's immediate surroundings. The path planning block also handles the vehicle-to-vehicle communication. In order to more accurately simulate this communication, it also incorporates communication delays/latencies. When communicating data from one vehicle to another, it generates a random non-zero number (within a specified range) indicating the delay. This delay value is decremented every subsequent time step and the data is only delivered to the intended vehicle once the delay value reaches 0.

4 DESIGN TRADE STUDIES

4.1 Shape of Track

The paper [**2**] that we wanted to model our project from used a two lane track consisting of concentric circles. However, from a path planning side, this required both longitudinal and lateral movements. Not only would we need a car-following model to simulate highway driving, but we would also need a lane-changing model in order for the vehicles to maneuver around obstacles safely. We decided to change this design into a figure-8 track where each set of cars only travels in their respective circle of the track. This would allow for the vehicles to follow a simple circular path, without changing lanes, and focus on a car-following/object detection model. When making decisions, each car would only have to change it speeds as opposed to both its speed and direction. The differences between the two tracks are shown below in Figure 4.



Design 1: Concentric circles track Design 2: Figure-8 track

Figure 4: Comparison of Proposed Track Designs

4.2 Car-Following Driver Model

With the new focus on the longitudinal movement, we had a couple car-following models to select from. We decided to use the Intelligent Driver Model instead of the Newell or Gipps's Model. The Gipps Model is a relatively simple model, developed in the 1970s and details the relationship between each vehicle's position, velocity, and braking severity. However, the Intelligent Driver Model is more recently developed, and its main purpose was to improve on Gipps's model, since the latter loses realistic properties in the deterministic limit. The paper [**2**] that attained similar goals to our project also makes use of this driver model, which can give us guidance on how to implement it. While tuning the parameters that define each vehicle's movements, we met various tradeoffs such as increasing max acceleration to increase throughput but decrease the efficiency of the system (harsher speeding up and braking).

4.3 Central Scheduler

To simulate vehicles communicating with one another in the cooperative case, we used a central scheduler that shares information to the vehicles. Data from the server is received and forwarded to all of the vehicles simultaneously as opposed to each vehicle having to make their own individual connection to every other vehicle. This ultimately avoids the need for distributed consensus and decision making between the vehicles. Scaled to the real world, this central scheduler behaves as a Roadside Unit (RSU), which is a wireless communication device that passes information, including traffic and safety warnings, to vehicles that pass along it on the road.

4.4 Multi-vehicle Passage

In the cooperative case, we allow multiple vehicles from the same track to pass through the intersection at a time. This is called car chaining. As cars share their intended path, including their current acceleration, to one another, vehicles in a chain can follow and accelerate at the same speed as the leading vehicle. This reduces delays from acceleration and deceleration in the trailing vehicles. The green circle in Figure 5 shows an example of car chaining. All three cars in the green circle will pass through the intersection before any cars on the right track will pass through. The cars within the orange circle are also in the same chain as the green cars, therefore, the orange cars also move at the same rate as the green cars without leaving gaps in between.

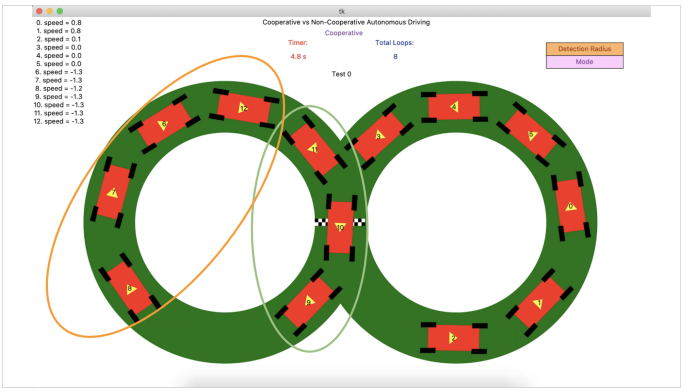


Figure 5: Example of Car Chaining

5 SYSTEM DESCRIPTION

5.1 Car-Following Model

The Intelligent Driver Model relates the positions and velocities of individual vehicles based on the the vehicles directly in front. The model defines the following notation [**2**]:

- v_0 is the velocity the vehicle drives at in free traffic,
- s_0 is the minimum following distance necessary between vehicles,
- T is the minimum possible time to the vehicle in front,
- a is the maximum vehicle acceleration,
- b is the comfortable braking deceleration (positive number),
- δ is the acceleration exponent (usually set at 4),
- x_α is the position of the front of vehicle α at time t,
- v_α is the speed of the vehicle α at time t, and
- l_α is the length of the vehicle.

The model defines the net distance between two vehicles as,

$$s_\alpha := x_{\alpha-1} - x_\alpha - l_{\alpha-1} \tag{1}$$

and the approaching rate between two vehicles as,

$$\Delta v_\alpha := v_\alpha - v_{\alpha-1}. \tag{2}$$

Using Equations 1 and 2, the model is able to define the function of acceleration of an individual vehicle as a function of time,

$$acc_\alpha = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}}{s_\alpha}\right)^2\right] \tag{3}$$

With the parameters mentioned in the Design Requirements section, we scaled and tested using these model's equations. Because the vehicles in our simulation are not robo-taxis with passengers, we are able to increase our maximum acceleration and have harsher braking in order to loosen our constraints slightly.

5.2 Information Constraints

As mentioned before, the server/path-planning component has a full view of the system and controls each vehicles' movement but needs to ensure that it limits a vehicle's view of the system appropriately.

For the **non-cooperative** case, we needed our constraints to accurately simulate what vehicles would "see" in a non-cooperative setting. We enforced a detection radius around each vehicle and when making its decision, only considered obstacles within that radius.

For the **cooperative** case, the information available to each vehicle is also bounded by some radius. But a key difference is that each vehicle also has information on the other vehicles' intended paths/acceleration, which provides a much more thorough picture of the moving parts in the system. This extra knowledge would be especially beneficial in a more complex track configuration such as a two lane track where a vehicle's intended path can also indicate which lane it plans to drive in, however, we are still able to experience its advantages on a figure-8 track as vehicles can communicate their acceleration to each other, enabling cars to chain together.

5.3 Scheduling Algorithm

Our goal for the scheduling algorithm was primarily to increase the throughput of the system but also to maintain fairness. We implemented scheduling polices that would increase the flow of the system whilst minimizing/avoiding the starvation of either lane of the track. To determine what lane to let cars through from, a priority is assigned to both lanes. The priority of a lane is calculated based on a weighted combinations of features of the state of the lane. The first feature we consider is the length of the queue. This information is communicated between vehicles and the front most vehicle of a lane would have a rough estimate of the length of the queue. In an attempt to increase fairness, we prioritize lanes with a longer queue so that we keep less cars waiting. The second feature we look at is the distance of the front most vehicle to the entrance of the intersection. In contrast to the first feature, we prioritize lanes with a smaller distance between their front-most vehicle and the intersection. This is done to increase throughput so that the vehicles closer to the intersection don't have to stop and wait for vehicles from the other lane (which are farther from the intersection) to pass through. Given this setup, we calculate the priority of a lane the following way:

$$priority = w_1 * f_1 + w_2 * 1/f_2 \tag{4}$$

In the equation above, f_1 is the length of the queue and f_2 is the distance of the front-most vehicle of the lane to the intersection. We take the inverse of this distance because we would want a lane with a smaller distance to be given a higher priority. One advantage of this scheduling method is its scalability. Since we use a weighted combination of features, we can easily scale this scheduler to more complex track configurations by simply incorporating more features. Lastly, we allow multiple cars to pass through the intersection from one lane. Doing this gets rid of delays involved with the frequent starting and stopping of the vehicles queued up at the intersection

Even with the described scheduling framework there is still a possibility for starvation. If one lane is much longer than the other, then even if we let cars through from that

lane, there still may be a lot of cars left in the queue and thus it is very likely that the same lane will again be prioritized by the scheduler. This situation is problematic because it starves out the other lane. To remedy this, we designed the scheduler to try to schedule cars from the other lane right after scheduling from one lane.This way we fairly distribute access to the intersection across lanes.

5.4 Non-cooperative mode

In the non-cooperative mode the vehicles do not communicate with each other and act solely based on knowledge about the obstacles within their detection radius. Given the car following model, vehicles are able to react to movement of their preceding vehicle. At the intersection, the car closest to the intersection entrance is given the right of way. If two cars from both lanes have the same distance to the intersection, the car on the left lane is given the right of way.

5.5 Cooperative mode

5.5.1 Queue-length communication protocol

The intersection scheduling algorithm relies on the front-most vehicle knowing the length of the queue on its lane. As a vehicle approaches the intersection, it informs the vehicle in front of it that there is one car behind it, and in the same way, that vehicle informs its preceding vehicle that there are two cars behind it. This information relay continues along the queue of cars and the length of the queue is incremented and propagated across the vehicles. This way the front-most vehicle has a rough estimate of the length of the queue on its lane. This information can then be sent to the scheduler to make a decision. Another thing we took into consideration when determining the length of the queue is the separation between the cars on the queue. If there are four cars on a lane and the first two cars are close together but the last two are far from the first two, it will be unfair to account for the last two cars when determining the length of the queue and possibly having to halt the cars on the other lane just so all the four cars from the first lane can pass through even though the last two cars are really far away. Because of this, when determining the length of the queue, we only take into account the length of the string cars where the separation between each of the cars in the string is less than some threshold. Thus a vehicle will only communicate the length of the queue to its preceding vehicle if it is close enough to it.

A decision is made by the scheduler only after the front-most vehicle is sufficiently close to intersection. If a decision is too early, it could potentially be made prematurely since the length of the queue could change in a short amount of time.

5.5.2 Car chaining

When a string of cars move, there are a lot of intermediate delays as it takes a while for the second vehicle in the string to react to movement of the lead vehicle, and in the same way it takes a while for the third vehicle to react to the movement of the second vehicle and so on. These are disadvantageous because it adds extra latency between vehicles moving as well as introducing empty space in the track when the vehicles do not move right away. As the length of the chain grows, these delays and space cascade and become a large source of the system's inefficiency. Because of this drawback, we introduced the concept of car chaining. With car chaining, the lead vehicle communicates its acceleration to all other vehicles in its chain so that every other vehicle is able to accelerate and decelerate at the same rate as the lead vehicle. This gets rid of the aformentioned delays involved in the movement of a chain of cars. We are able to implement this in our cooperative framework because vehicles can now communicate their intended acceleration and deceleration to all the vehicles behind it in the chain. Therefore, if the lead vehicle needs to brake harshly to avoid collision, all vehicles in the chain can do so at the same time.

6 Testing

To test the performance of our solution, we wrote a script to create randomly generated 50 test cases with between 9-13 cars in each test case. Each test ran in both the cooperative and non-cooperative modes for 40 seconds and outputted metrics for both cases. The acceleration and deceleration values were summed up for all vehicles and averaged across all the vehicles. We did the same for the acceleration and deceleration frequencies, that is, every time the acceleration values were positive or negative, we incremented acceleration and deceleration counts respectively.

Initially, we ran the tests based on system time but realized that it would be much more efficient to decouple from system time. This would allow us to remove the graphics component from the testing and run the iterations much faster than the actual time allotted. Testing using simulation time would keep the number of control loops the same across tests, but run it much faster with respect to system time. We set each iteration to be the equivalent of 0.015 seconds of system time, so to run a 40 second test, we needed to run $40/0.015 = 2666$ iterations. But our computers were able to run these orders of magnitude faster than 40 seconds. This allowed us to run many more tests at a time, aggregate those results (i.e. total throughput, average velocity etc) and write them to csv file.

7 Results & Observations

Results

- **30.33%** increase in throughput in cooperative vs non-cooperative case

	Non-Cooperative	Cooperative	% Change
Avg Velocity	0.094 m/s	0.121 m/s	↑ 28.23%
Avg Acceleration	0.0024 m/s^2	0.0032 m/s^2	↑ 31.25%
Avg Deceleration	0.024 m/s^2	0.043 m/s^2	↑ 77.08%
Avg Acceleration Counts	515.77	195.22	↓ 62.15%
Avg Deceleration Counts	1497.5	1463.3	↓ 1.29%
Avg Waiting Time	8.79 s	4.93 s	↓ 43.88%
Total Loops	2305	3004	↑30.33%

Figure 6: Test results

We saw a 30.33% increase in the throughput of the cooperative system over the non-cooperative system. We also saw a similar increase in the average velocity in the system which was expected as a higher velocity signifies more flow in the system. Conversely, we experienced a decrease in waiting time, which we also expected would go hand in hand with an increase in throughput. One metric that we did not anticipate was seeing the average acceleration and deceleration values increase in the cooperative case.

7.1 Lower waiting time

A number of factors contribute to the lower waiting times experienced in the cooperative case. Firstly, the car chaining mechanism allow vehicles to react faster to the movement of the lead vehicle. Because of this, the vehicles spend less time waiting for the preceding vehicle to move far enough from it so it can move. Once the lead vehicle begins moving, all vehicles in the chain become aware of its motion and can advance early.

Secondly, the intersection scheduler prioritizes lanes with longer queues and this policy minimizes waiting times because it stops the lane with fewer cars instead of the other longer lane which would translate to more vehicles waiting on average.

Lastly, we allow multiple vehicles to pass the intersection. This gets rid of the need for vehicles queued up at the intersection to start and stop frequently as they approach the entrance to the intersection.

7.2 Higher acceleration/deceleration values

We initially expected to see lower acceleration and deceleration values in the cooperative case but these values actually increased instead. Further analysis allowed us to deduce the reason for this. As we mentioned before, multiple vehicles are allowed past the intersection in the cooperative case and this reduced the frequency with which vehicles queued up at the intersection had to start and stop repeatedly as they pushed towards the entrance of the intersection. Instead, vehicles move collectively and are able to reach their max velocity much more often. This results in overall higher acceleration values because there is a higher change in velocity over time. However, we still theorized that the cooperative approach increased the overall efficiency of the system. We deduced that this translated to less frequent acceleration/deceleration overall in the system. We tested this theory by examining the acceleration/deceleration frequencies and found them to decrease as we expected.

7.3 Cooperative mode is more advantageous with denser car populations

As we increased the number of vehicles present in the test cases we noticed more improvements in the cooperative case over the non-cooperative case. This came as no surprise to us as a denser track layout means more contention for the intersection, allowing the advantages of cooperative driving to really take effect. With a combination of the car chaining and multi-vehicle passage, the intermediate delays in the system are mitigated and traffic flow is boosted.

8 PROJECT MANAGEMENT

8.1 Schedule

Refer to Figures 7 for our project's schedule. It reflects the the design changes made due to the shift in implementation. As you can see, we split up the work evenly and worked in parallel. In an attempt to reduce the integration time, we first laid out a plan for the structure of our code and designed our interfaces before hand. This way it was easy to connect all the modules together. Nevertheless, we set out times in our schedule to integrate along the way in order to mitigate risks.

8.2 Team Member Responsibilities

After switching solution approaches from a physical demo to a software simulation, we broke down our tasks into three parts: graphic simulation, cooperative framework, and Intelligent Driver Model. Serris worked on designing and creating the graphic simulation. This also involved writing code to collect and aggregate the results/metrics from the simulation.

Tito worked on the cooperative framework which involved the vehicle to vehicle communication and the intersection scheduling algorithm.

Lastly, Kylee was in charge of the Intelligent Driver Model for car movement. He was also in charge of implementing the car chaining mechanism and creating the script that randomly generated test cases.

A major part of this project was testing and figuring out ways to improve the throughput in the system. We all ran our own tests individually to assess/analyze the performance of our solution and look for any optimizations that we could make. After running multiple tests, we collectively discussed potential changes that we could make to increase throughput.

8.3 Budget

Refer to Table 2 for our project's budget. This includes components we have purchased before our change in design. All of the parts that were purchased were initially for our physical demo implementation, which included hardware components. Since our design change resulted in our project being completely software-based, no more purchases were made.

8.4 Risk Management

Our project involved many components which none of us had much experience with. Due to the physical limitations, we decided to shift our implementation to become purely software-based. This allowed us to continue to focus on our main goal: simulate and measure the effect between cooperative and non-cooperative autonomous driving. In addition, this mitigated our budget risk as no more purchases were made, and collaboration on the project was smoother and less complicated.

We realized our solution approach was very reliant on the information that was being shared between the vehicles. This is dependent on the decision making process for each vehicle and track setup. After researching many scenarios, we decided to look into two track setups. One of them was a figure-8 track with circles of equal radius and the other was a two-lane circular track. With lane changing, the latter made the path planning algorithm more complicated as it involved additional driving models. The former approach simplified the problem as it restricted the movement of the vehicles to only accelerate and decelerate. As a result, we started implementing on the figure-8 track to generate performance metrics.

To ensure we would meet our requirements, we began testing early on. Doing so allowed us to discover problems which allowed us to make the necessary design changes. For example, we noticed it was difficult to get performance metrics without being familiar with the behaviors of cars individually and of their possible interactions with one another. As a result, before generating test cases, we designed a customizable setup, where users can click where on the track they want to insert vehicles and visualize their movement. These graphics allowed us to reconsider our design for what and how we were generating test cases in order to meet our requirements. We took into account the possibility of our manual test cases being biased or skewed to meet our specifics so we randomly generated 50 test cases to provide more fair and accurate results.

9 Future Work

9.0.1 Two-Lane Track

While working with a single lane figure-8 track simplified our problem, it also limited the benefits/advantages of the cooperative driving framework. Vehicle to vehicle communication extends each vehicle's sensory range and allows it to plan ahead. In a single lane track, the extra knowledge isn't as useful because there is only so much that a vehicle can do - either starting or stopping. A two-lane track design would potentially see many more advantages of vehicle to vehicle communication as vehicles could warn other vehicles of any obstacles in the road, allowing them to switch lanes earlier to avoid the obstacle without causing more traffic.

With v2v communication, vehicles could also change lanes more smoothly as they could communicate their intended path with others. When switching lanes, a vehicle could inform other vehicles in its vicinity its intended position (after switching lanes) and all such vehicles could adjust their speed to accommodate for the lane change.

9.0.2 Physical Demo

The next step from a software simulation would naturally be a physical demo using robotic cars. With the physical demo approach, we envision having multiple robotic cars each equipped with a micro-controller that is able to communicate with other devices either through WiFi or Bluetooth. As we mentioned earlier, the intersection scheduler is a centralized unit which has all the information on the system and makes a single global decision for all the vehicles. Implementing this in a physical demo would involve having a server act as a Roadside Unit (RSU) that collects information from the vehicles and distributes decisions to all vehicles. The Intelligent Driver Model also works for a physical demo but the vehicles would need cameras/sensors to be able to detect obstacles. Without sensors for every vehicle, the server could also be equipped with a global camera that would detect all the vehicles in the system and control each based on its immediate surroundings.

10 RELATED WORK

The Prorok lab supervised by a Cambridge University professor, Amanda Prorok, also worked on a similar project

[**1**]. The project involved an experimental testbed consisting of 16 miniature RC vehicles. Their experiment was executed on a multi-lane track with an obstacle placed on one of the lanes. The demonstration showed how the communication between vehicles could prevent a buildup of traffic on the blocked lane. Their primary goal was on safety as they tested the fleet in both driving modes with normal and aggressive driving behaviors. In their experiments, from non-cooperative to cooperative driving, they were able to see a 35% improvement of traffic flow with normal driving and a 45% improvement with aggressive driving.

11 SUMMARY

From our initial solution approach, our project has undergone many design changes. However, they are a result of refining our design requirements and simplifying the problem to focus on our main goal: comparing the effects between cooperative and non-cooperative autonomous driving. To get an accurate measurement of our system's performance, we immediately began implementing and testing our designs to make the necessary design changes early on. From what we have encountered, we have been able to address most of our design challenges with concrete metrics through testing or potential alternatives as we continue to experiment. With a 30.33% increase in throughput between the cooperative vs non-cooperative mode, we believe we have shown the significant advantages in performance when vehicles are able to communicate important information about their decisions. Our results highlight the benefits of pursuing further research in this field, and we hope to see more progress as autonomous vehicles begin to integrate with our daily lives.

References

[1] Nicholas Hyldmar, Yijun He, Amanda Prorok. *A Fleet of Miniature Cars for Experiments in Cooperative Driving.* Paper presented at the International Conference on Robotics and Automation (ICRA 2019). Montréal, Canada, 2019.

[2] Treiber, Martin, Hennecke, Ansgar, Helbing, Dirk. "Congested traffic states in empirical observations and microscopic simulations", Physical Review E, 62 (2): 1805–1824. 2000.

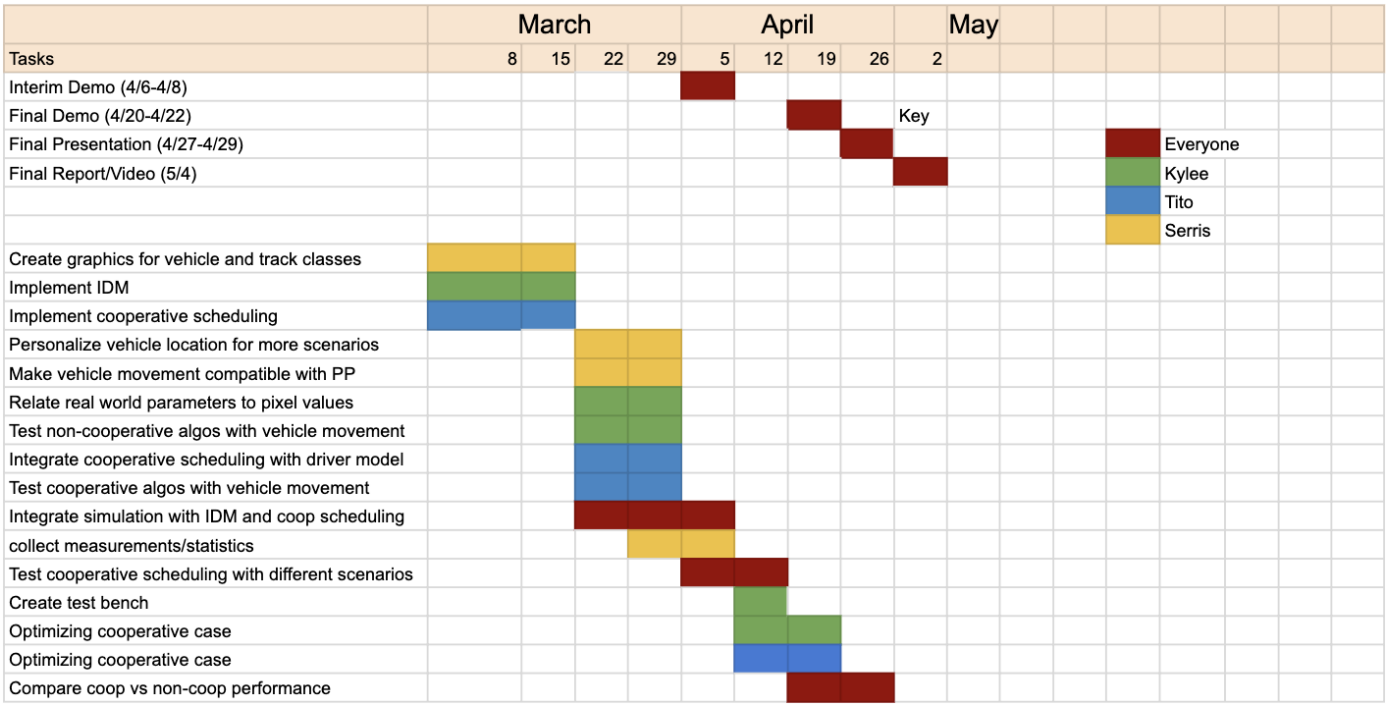


Figure 7: Updated Gantt Chart after design changes

Purchased			
Item	Quantity per Order	Order Quantity	Total Price
Mini Robot Chassis Kit	1	1	29.95
L293D IC	10	1	8.00
DC Motors + Wheels	4	1	14.59
Roller Ball	1	1	5.68
Mini Breadboard	6	2	13.96
NodeMCU	3	3	38.97
9 Volt Batteries	8	1	10.99
Logitech HD Pro Webcam C290	1	1	59.99
Logitech Webcam Mount	1	1	19.98
L298N Motor Controllers	5	1	13.99
On and Off Switches	12	1	9.88
TOTAL			**225.98**

Used from Lab
Item
Jumper Wires
Battery Clips

Total Budget	
State	Price
Purchased	225.98
TOTAL	**225.98**

Table 2: Budget of tools needed for the project