Final Challenge

Team # 2

Isaac Lau
Lilly Papalia
Mario Peraza
Joshua Rapoport

6.141/16.405: RSS

May 21, 2021

1 Introduction

Author: Mario Peraza

Throughout the course of RSS, our team had developed and implemented multiple types of controllers and algorithms to accomplish tasks. All the information gained throughout the semester was put to use in the final challenge. This challenge was composed of two parts. The first being the final race where teams would compete to see whose car simulation could make it around the course in the fastest time possible with the least number of collisions using pure pursuit path following to follow a predetermined path of our design. The second part was an obstacle course where 10 objects would be placed at random on a straight track and the goal was for the car to complete the course as fast as possible with little to no collisions using the lidar sensor data.

2 Technical Approach

2.1 Challenge 1: Final Race

The final race track was a figure eight type shape which drove around buildings, through narrow passageways, and ended where it started (Figure 1).

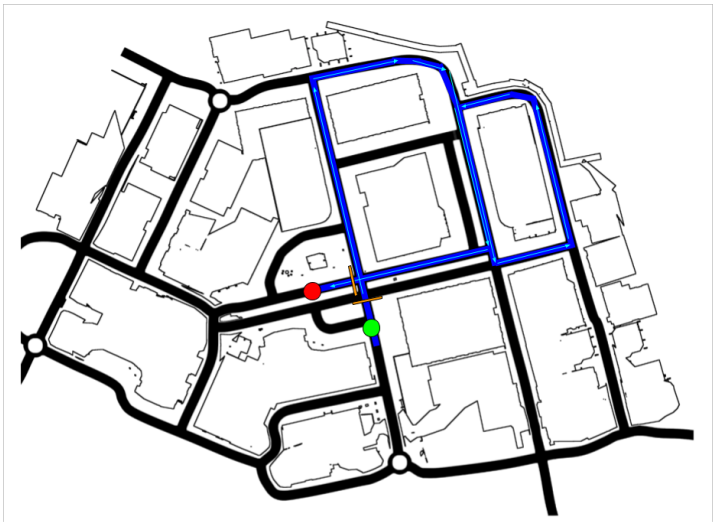

Figure 1: Map showing the final race track from start (green) to finish (red).

2.1.1 Building the Trajectory

Author: Isaac Lau

In building a path for the final race trajectory, safety was the primary priority. As a result, the trajectory optimization / building process started with a simple path that allowed the race car to navigate through the course with the least risk possible. After a base line had been established, the speed and path routing was optimized to improve track time. In particular, the 2 main tuning parameters for this process was the target speed for the pure pursuit controller and the specific cornering strategy that the race car employed. With regards to cornering, the final strategy that was deployed in the final challenge was that the race car entered turns slightly wider than normal and then "apexed" closely around each corner to minimize time spent cornering. [Figure 2] As part of this routing optimization, the race car would also avoid clipping the curb since this perturbation would introduce instability within the race car's path and lead to a deviation from the target path afterwards. Ultimately though trial and error testing, a set of 22 trajectory points were chosen and published for the race car to follow.

Figure 2: Cornering strategy of entering turns slightly wide and "apexing" around the corner point to minimize distance / time spent turning

2.1.2 Pure Pursuit

Author: Joshua Rapoport

The pure pursuit algorithm (see Lab 6: Path Planning), offered the robustness and functionality required for the final race (and eventually the obstacle track as well). An early hurdle was to modify the algorithm to accept trajectories that loop and intersect on itself.

Base Algorithm

1. Identify the closest point p_{close} on the trajectory to the robot. This was done by calculating the closest point on every segment of the trajectory, then selecting the point on segment i_{close} corresponding to the minimum distance.
2. Define a transient goal p_{goal} point that exists on the trajectory a distance $|p_{close} - p_r|$ away from the robot. (Only solutions after segment i_{close} are considered valid).
3. Calculate a steering angle δ that will, given no friction or drifting, guide an Ackermann drive-controlled robot to p_{goal}.

Upon the robot's odometry being updated (in this case, by the "ground truth" localization), a callback is made to the trajectory follower:

```
def odom_cb(msg):
    p_robot, heading = pose_from_msg(msg.pose)

    # 1. Closest Trajectory Point
    p_close, i_close = get_closest_point(p_robot)

    # 2. Transient Goal Position
    p_goal = get_transient_goal(i_close)

    # 3. Pure Pursuit
    turning_angle = evaluate(p_goal)

    # Publish drive instructions (speed, turning_angle)
```

We will reuse the parameters for wheelbase length L, look ahead distance d, and velocity V_{max} as defined in Lab 6.

2.1.3 Modifications

In order to successfully follow the path describes in Figure 1, the algorithm above required some modifications - in particular, a more robust selection of p_{close}, and a damper on the steering angle.

Loops and Intersections As it was, the pure pursuit controller from Lab 6 only kept track of the current closest segment i_{close}. The logic that selected p_{close}, i_{close} fails when two segments are within a lookahead distance of the robot. This allowed for paths that loop on itself, intersect with itself, or otherwise have two non-adjacent

Cross-Platform Stability Recall from Lab 6 that the majority of development for the pure pursuit controller was done in the provided 2-dimensional simulation, which does not account for vehicle tilt, friction, or drifting. When transitioning to the 3D TESSE simulator for the final race, we noted extreme oscillations in the robot's heading. Although successfully staying withing 1 meter of the loaded trajectory, the oscillations were so extreme that the robot's top speed capped at approximately 2 meters per second - a speed that would have resulted in a DNF time.

A very cheap solution was found in the form of a proportional controller. Through rigorous tuning, we found stable configurations for our robot ranging from low speed (1 m/s and 30% of steering angle) to high speed (13.5 m/s and 4% of steering angle). This had the added benefit of damping the system response *without* modifying the control logic of the original steering angle.

Although we observed reduced control authority on high-speed turns, this consequence could be mitigated by robust trajectory building methods (methods described above).

2.2 Challenge 2: Obstacle Avoidance

The obstacle avoidance course was a straightaway where obstacles would be randomly places along it (Figure 3).

Figure 3: Map showing the start (right) and finish (left) points of the obstacle avoidance course with example trajectory in red.

2.2.1 Lidar

Author: Lilly Papalia

The goal for the obstacle avoidance challenge was to not hit obstacles, while moving as fast as possible. The initial focus was on the actual obstacle avoidance, and to do this the lidar scans were used. The lidar scan is split into sections depending on the angle, as shown in Figure 4. The red lines show the approximate range for the front of the car, while the green lines show the cut off for the sides of the car. Since the car is moving forward, only scan values in the front half to the car are observed.

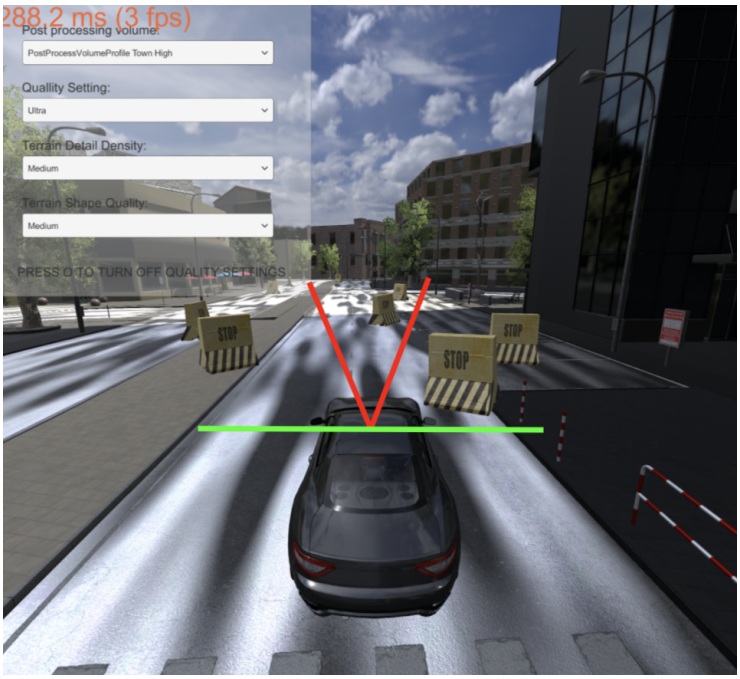

Figure 4: Diagram showing the approximate angles used to split the lidar scan. The red lines show the range included as the front of the car, while the green shows the cut off for the sides of the car.

The distances directly in front of the car are checked to see if there are any obstacles within the threshold distance. If the minimum value of the scan in this range was greater than the threshold distance, the car continues driving straight.

If there is an obstacle in front of the car, the minimum distances on both sides of the car are calculated. The two minimum values are compared, and the car chooses to turn the direction with the larger minimum distance. Another check is if the front distance is very small, the car publishes a negative velocity to help avoid hitting the obstacle, and if there is a collision, the car can self correct and continue down the path.

2.2.2 Iterations

Author:Mario Peraza

Although the car could successfully avoid obstacles using only the lidar data, it was not able to make it to the end of the track due to the lack of steering control within our system. To correct this mistake multiple iterations of the obstacle avoidance controller were developed and tested to assure the car could

complete the course in an efficient manner.

Line Follower The first attempt made to correct the obstacle avoidance controller was implementing a line following mechanism. Two points were taken from the map, a start point and an end point. Once found, an equation for a line was determined using the world coordinate system. After this line was created, the cars x position was taken and input into the line equation to determine whether the car should turn left or right depending on what side of the line the car was on (Figure 5).

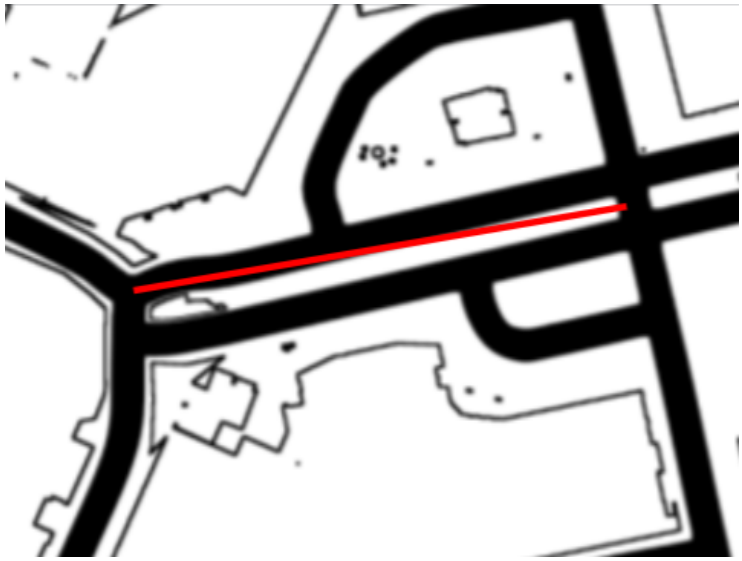

Figure 5: Map of obstacle avoidance course showing the red line which the first iteration of obstacle avoidance followed.

Once implemented this approach showed unexpected results with almost random car movements. This may have been due to the lag time between the cars position which was passed into the line equation and where the car currently was. If the cars position was updating faster than being passed into the line equation, then the movement would not reflect what expected.

Edge Detection The second version of obstacle avoidance utilized the edges of the course to correct the cars path. Once the edges were mapped, the cars position would constantly be checked to determine if the car went out of bounds. If the car were to drive out of bounds, the steering angle would correct itself to drive the car back onto the course.To determine if the car was out of bounds, the current y position of the car was checked to see if it was above or below the two edges (Figure 6).

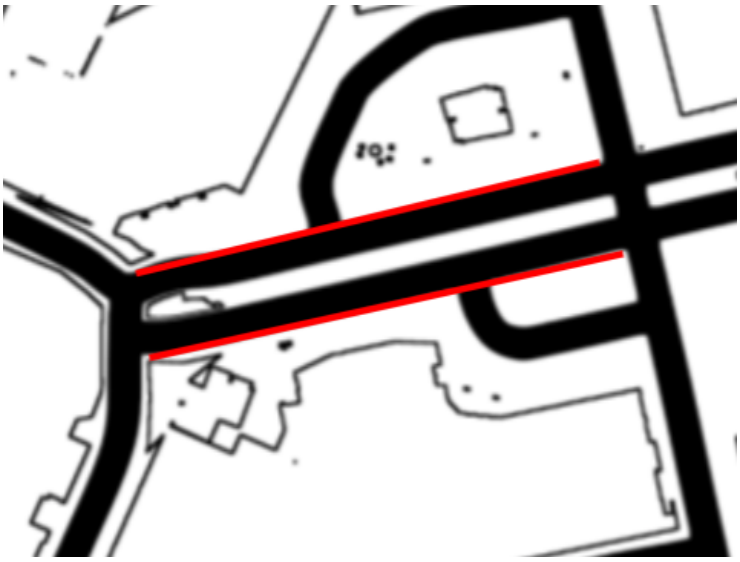

Figure 6: Map of obstacle avoidance course showing the right edge (top) and left edge (bottom) of the course in red.

This approach allowed for some test runs to successfully complete the course. However, there were times where the car would not correct itself fast enough to turn back onto the course and get stuck on a wall or other obstacle along the edge of the course. In an attempt to correct this, a buffer was added to the edge of the wall in hopes that the car would turn earlier. This caused other malfunctions where the car sometimes became stuck out of bounds, only turning in one direction.

Edge Deflection The third approach utilized the edges in a different way. Instead of waiting for the car to be out of bounds, the car would now continuously turn away from the edge that it was closer to. The distance was taken between the cars current position and the closest point of each edge. Whichever distance was shortest meant that the car was closer to that edge, resulting with the steering angle turning away from that edge. This method was similar to the line following method, however now the car would follow an imaginary center line (Figure 7).

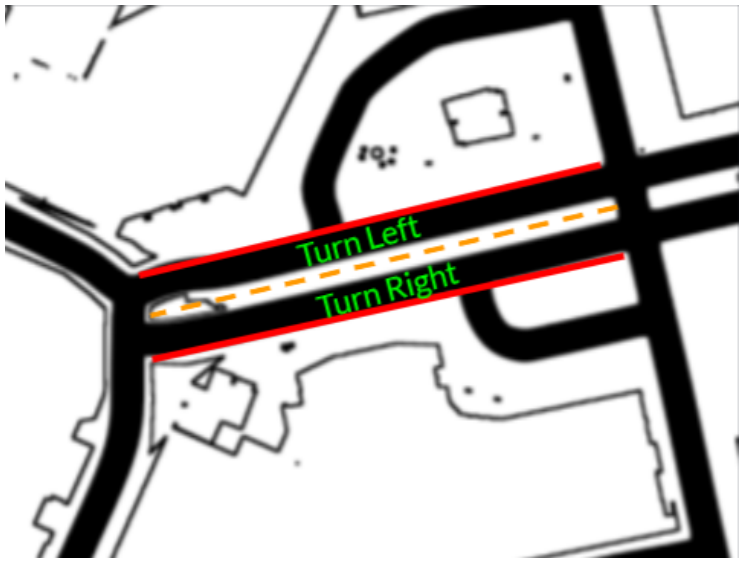

Figure 7: Map od obstacle avoidance course showing the edges of the course and the imaginary center line in orange.

Edge deflection proved to be a viable approach as multiple test runs were able to successfully complete the course. One drawback was that the car had large uncontrollable oscillations due to the continuous turning of the steering direction. At points, the car would over correct itself while simultaneously avoiding an obstacle which caused the car to completely turn around and start driving in the wrong direction.

Pure Pursuit For the fourth and final approach, the pure pursuit model our team had developed was integrated into obstacle avoidance. A goal point was chosen at the end of the course and fed into the pure pursuit controller. This enabled the car to constantly drive towards the desired end goal. When the car detected and object, it would stop using pure pursuit to avoid and then continue to adjust to the end goal. Overall, this final version worked the best, rarely crashing into objects while going at high speeds.

3 Experimental Evaluation

3.1 Final Race

Author: Isaac Lau
In evaluating the results of our final race code and trajectory independent of the final competition, the race car was tested against track conditions across 10 different experimental trials. All 10 trial runs were 100% collision free. From race day, the times to complete the track were 58.88, 59.04, and 58.88 seconds

respectively. Likewise, the race car was extremely repeatable with a deviation of +/- 0.5 seconds across all trials. This lent credence to our team's philosophy of safety first and having as little variance as possible in the race car's race trajectory even with the risks that taken to increase track time. In a more detailed analysis of Figure 8, a close inspection of the right hand side of the 2nd loop reveals a short period of low frequency oscillations. Figure 9 shows the published steering angle over time. The small oscillations can be seen, while the large spikes are the turns on the race course. While the race car originally suffered from a high frequency oscillation effect early in the tuning process, adjusting the constants of the PD controller helped to mitigate this effect. Note that given the relatively short nature of that section of the track, this minor deduction in performance did not heavily affect the race car's track time. A video of the three trials from race day can be accessed here

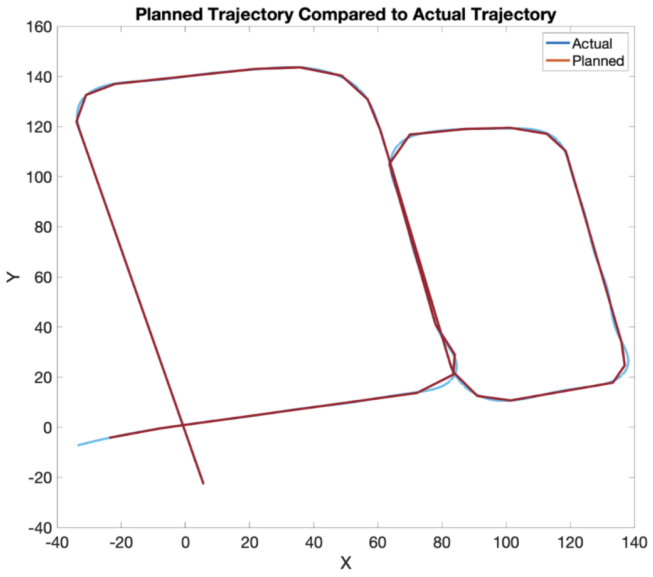

Figure 8: The planned trajectory compared to the actual trajectory of the car. The car follows the trajectory very closely, and the only deviation is at the turns.

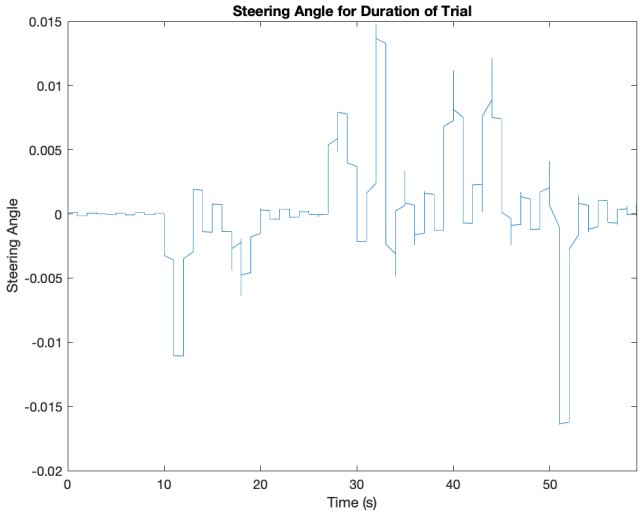

Figure 9: The published steering angle over the course of a trial, at the beginning there is very little oscillations, and as the trail continues the oscillations increase. The large spikes are the turns on the course.

3.2 Obstacle Avoidance

Author: Lilly Papalia

For the experimental evaluation of the obstacle avoidance, ten trials for five, ten, twelve, and fifteen obstacles were completed. This is a total of forty trials of obstacle avoidance. In order to monitor the number of collisions, the /tesse/-collision node was echoed, as well as visually observing the car in rviz. Out of these forty trials, twenty or fifty percent did not collide with any obstacles. The max number of collisions was 3 and this only occurred in a total of in 3 trials. There were also 3 trials classified as did not finish because the car was stuck from the collision, and could not recover in these three cases.

As shown in Figure 10, the car performed better with lower number of obstacles. Our approach performs very well for ten obstacles, averaging 0.3 collisions per trials. Increasing the number of obstacles from ten to twelve resulted in an average increase in collisions of 0.9. The implementation does work for high numbers of obstacles, but it is much less consistent. If the car does have collisions, in general it is able to recover and make it to the end of the course. Further tuning of the implementation would improve the performance on higher numbers of obstacles.

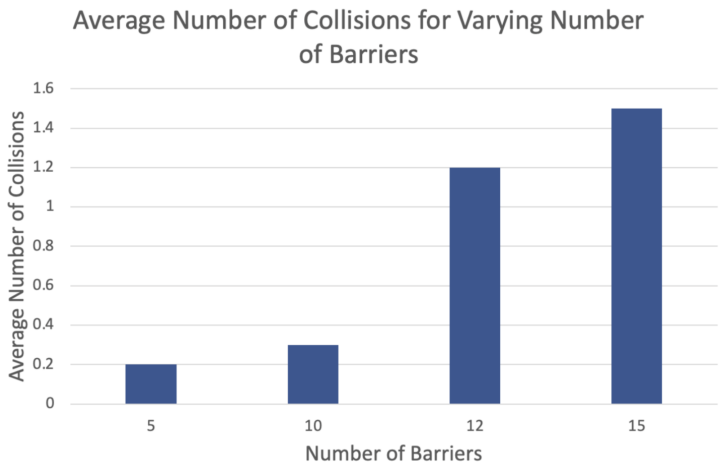

Figure 10: Bar graph displaying the average number of collisions over ten trials for each number of obstacles. As shown in the graph, the implementation works very well for ten obstacles or less, averaging 0.3 collisions per attempt. Over 10 obstacles, the car sees an increase in the number of collisions.

In the code, a consistent 13 meter/sec speed is published. The course is approximately 100m long, and the fastest time was 7.06 seconds from race day. This translates to a speed of approximately 14 meters/second, a little faster than our published speed. The car is able to avoid obstacles while traveling at a fast speed.

A video of the three trials on race day can be accessed here. All three obstacle avoidance attempts had zero collisions, and the times were 7.30, 7.24, 7.06 seconds in that order.

4 Conclusion

Author: Isaac Lau
The augmented pure pursuit model and re-designed obstacle avoidance model were successfully integrated into the final challenge 3D environment. The improved pure pursuit model was able to navigate crisscrossing paths while also sporting adaptive speed control to accurately and efficiently traverse the final race car track. Likewise, the refactored obstacle avoidance model was designed and tested allow the race car to travel from a given start point to a given end point while also minimizing collisions with obstacles strewn about the obstacle course. Both parts of the final challenge built off of existing code bases from previous RSS labs but featured code augmentations and new tuning parameters for the specific challenges of this final lab. In the future, the algorithms and models which were developed in RSS can be scaled and tuned to be useful in other environments and help develop autonomous machines. Overall, Team 2 was quite please with the final race and the performance of the autonomous race car both in terms of speed and collision- free races.

Special thanks to the TAs (especially Valerie and Tanya), Professor Carlone, Jane, and the technical / communications teaching staff for their support throughout this semester!

5 Lessons Learned

5.1 Joshua

The final challenge was incredibly taxing and rewarding. We continued our balance of parallelization of work and pair collaboration. This lead to a successful implementation for both challenges

Our communication continued to improve, and we are happy with our final results!

5.2 Lilly

The final challenge was a great way to end RSS this semester! The two challenges utilized many of the skills we utilized throughout the previous six labs. I learned that iteration, and discussing ideas with teammates is what leads to success. Mario and I focused on the obstacle avoidance challenge, and discussing different ideas, as well as combining our different ideas is what lead to our final successful implementation. I also learned how to useful Github can be while changing ideas for the challenge. Being able to create multiple branches allowed us to try many different ideas without losing progress on the old ideas.

Using the VDI was very helpful and ran much faster and smoother than our local machines. This made is easier to iterate and test our ideas. In the end,

collaboration is what lead to our success in both the final race and obstacle avoidance.

5.3 Mario

The final challenge proved to have many difficulties to overcome. It was through lots of teamwork and testing that enabled us to succeed. I learned that it takes many trials and errors to determine how to best approach a situation. I was able to work with Lilly as we bounced ideas back and forth until one of those ideas was perfect for the job. In the end it was a combination of her and my effort which allowed our obstacle avoidance controller to be as robust as it was.

I also learned a lot about using Github this week. There were many different ideas and versions of the obstacle avoidance controller, which allowed me to learn to utilize branches quite often. As soon as we realized that we had started completely changing our designs, we made sure to create a new branch in case our new design was less successful than the previous. There was also a lot more communication that went into designing this week than previously, however we were able to test rapidly due to the VDI system which worked much better than our local systems.

5.4 Isaac

In terms of technical takeaways, many of the main ideas we used in this final challenge was drawn from previous labs. This taught me the importance of writing generalized code to allow for future, modular changes more easily and to avoid having to write large amounts of code from scratch. Effectively, since we luckily had a modular code base our primary task was to augment and tune code we had already written to fit the particular parameters outlined by this final challenge.

In terms of communication takeaways, I learned how to extensively use Github to keep track of different modules, pull old modules into components of our final code base, and submit push / pull requests between teammates while also engaging in open conversations so that we would be able to collaborate effectively without overwriting each other's work. This ultimately allowed us to achieve the highest possible score while also building off the code from each other, allowing us to iterate faster than we had ever been able to do previously.