Final Challenge Report: Race and Obstacle Avoidance

Team 12

Robert Williamson
Maria Pau Barbosa
George Chen
Lori Zhang
Kris Vu

6.141/16.405: Robotics: Science and Systems

May 21, 2021

Contents

1 Introduction **2**

2 Technical Approach **3**
2.1 Final Race . . . 3
2.1.1 Pure Pursuit . . . 3
2.1.2 Finding the Closest Trajectory Segment . . . 3
2.1.3 Calculating a Target Point . . . 4
2.1.4 Steering Control . . . 5
2.1.5 Velocity Control . . . 5
2.1.6 Custom Trajectory . . . 6
2.2 Obstacle Avoidance . . . 6
2.2.1 Empty Space Controller . . . 7
2.2.2 Different Approaches . . . 7

3 Experimental Evaluation **8**
3.1 Final Race . . . 8
3.2 Obstacle Avoidance . . . 8

4 Conclusion **10**

5 Lessons Learned **11**
5.1 Be Flexible to Changes in Approach (Pau) 11
5.2 Teamwork Makes the Dream Work (Pau) 11
5.3 Time management is key to success (Rob) 11
5.4 Small changes have big impacts (Rob) 11
5.5 Help everyone do what they like(Kris) 11
5.6 Big learning starts with small learning (Kris) 12
5.7 Mustaches are Cool (George) 12
5.8 I Love Robots (George) 12
5.9 (Lori) 12
5.10 (Lori) 12

1 Introduction

Author: Lori Zhang

As this semester comes to a close, the final challenge was the final step along the autonomous vehicle pipeline. Success in the final challenge was closely linked to our team's performance in the earlier steps in the pipeline. In Lab 1, we familiarized ourselves with the frameworks and tools necessary for the process, namely ROS and GitHub. In Labs 2 and 3, we learned how to read LIDAR data and built a controller capable of following a wall. In Lab 4, we implemented object detection algorithms and performed real-time coordinate transformations. between the image and world frames. In Lab 5, we wrote scripts capable of performing robot localization in both 2D and 3D frames. Finally, in Lab 6, we implemented large-scale path planning and a pure pursuit controller able to follow trajectories accurately and quickly. As we had taken care to test and validate the performance of these components, we had established a strong foundation for our final challenge tasks.

The final challenge consisted of two main modules: the final race and obstacle avoidance. For the final race, we were able to rely upon our pure pursuit controller framework from Lab 6 and adapt it for this purpose by tuning controller hyperparameters as well as adding a velocity controller to maximize robot speed on the straightaways and slow down through the corners, ensuring fast and collision-free autonomous driving. Some modification were also made to the steering controller as well to account for these constraints.

However, for real-time obstacle avoidance, we had to implement a new controller. This is due to our choice of path planning algorithm in Lab 6 – A*, though a relatively fast and accurate search-based algorithm, was not fast enough for the real-time obstacle avoidance task. Initially, we had attempted to implement an incremental RRT path planning module, but due to time constraints, eventually switched to an empty space controller publishing steering directions to our robot in response to LIDAR data.

In the end, our robot completed the final race in 44.6 seconds with no collisions and won the virtual race. Our robot also completed the obstacle avoidance track in 9.85 seconds with 2 collisions. Overall, by combining our previous work in implementing real time robotics software, reading and transforming sensor data, and writing algorithms that use this data to make behavioral decisions, we were able to not only succeed in the final challenge, but also understand and build the autonomous vehicle pipeline by ourselves.

2 Technical Approach

2.1 Final Race

Author: Robert Williamson

2.1.1 Pure Pursuit

A modified pure pursuit controller was used to follow the custom built trajectory. The pure pursuit controller functions by calculating a target point a certain look-ahead distance away from the robot on the trajectory and steers towards it. Every time step, the target point is updated until the robot reaches the end of the trajectory such that the robot is always chasing a target point along the path. There are three main components of the pure pursuit algorithm: finding which segment of the path the robot is on or closest to, calculating the target point on that segment, and steering towards the target point.

2.1.2 Finding the Closest Trajectory Segment

The first step to chasing a target point is to find out where on the trajectory the robot is. The trajectory is imported to the pure pursuit function as a list of points, which can be broken into pairs of start points and endpoints where each pair of points denotes a segment of the trajectory. To know which segment of the trajectory the robot is on, a global counter is updated to reflect the highest index of the trajectory that has been completed. The robot starts on the first segment in the trajectory and once it has completed a segment, the global index is updated to the next segment. While this method would not work well in an application where paths are frequently planned, this simple solution increases the speed of the time pure pursuit loop. Run time is especially important in this race because the car is moving much faster than previous applications and drive commands must be published quickly enough to keep up with the robot's real-time location. The index is updated when calculating the target point.

2.1.3 Calculating a Target Point

The next step is to calculate the target point that the robot should chase. The target point that the robot chases is always the same look-ahead distance away determined by d_l which is a tunable parameter, in this case, $d_l = 22.5$m. To find the target point, a circle with radius d_l is drawn around the robot and then we check to see if there are any points on the current segment that intersect with the circle as shown in Figure 2 where p_1, p_2, r, and q are the segment start point, endpoint, d_l, and robot position respectively.

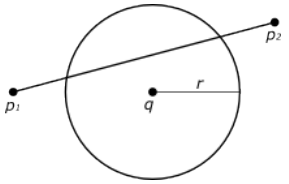

Figure 1: Diagram of the architecture used to calculate the target point.

The segment can be parametrized yielding:

$$p_1 + t(p_2 - p_1) \text{ with } t \in [0, 1] \tag{1}$$

A point p_{target} on the circle must satisfy $|p_{target} - q| = d_l$ to lie on the circle. First, however, the segment must intersect the circle which only happens when:

$$|p_1 + t(p_2 - p_1) - q|^2 = d_l^2 \tag{2}$$

Using $v = (p_2 - p_1)$ and dot product properties, the above equation expands to yield a quadratic function with respect to t:

$$t^2 \cdot (v \cdot v) + t(2(v \cdot (p_1 - q)) + (p_1 \cdot p_1 + q \cdot q - 2p_1 \cdot q - d_l^2) = 0 \tag{3}$$

Using the form $at^2 + bt + c = 0$ gives $a = v \cdot v, b = (2(v \cdot (p_1 - q))$, and $c = (p_1 \cdot p_1 + q \cdot q - 2p_1 \cdot q - d_l^2)$. The value of the discriminant of this function $b^2 - 4ac$ holds useful information about the intersection of the segment and circle. In this application, there are three cases:

1. $d < 0$ means that the segment does not intersect the circle at all. In this case, the current segment is discarded and the next segment is examined. This process ensures the controller always finds a target point that helps deal with sharp corners, short segments, or any other edge cases.

2. d ≥ 0 means that there is a real solution to $t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. In this case, only the larger of the two t values is considered as the robot wants to continue moving forward along the trajectory.

3. $t \notin [0, 1]$ means that the target point intersection point is not on the segment and the robot continues to the next segment to search for intersection points. This case occurs at corners and informs the robot to start evaluating the next segment of the trajectory. When $t > 1$ the global segment index is updated as this current segment has been completed.

Starting on the segment obtained from the closest point module, the robot iterates through the following segments until it finds a valid target point. The target point is simply $p_{target} = p_1 + t(v)$ which is returned in world coordinates.

2.1.4 Steering Control

Now that the target point has been calculated, the steering angle can be determined. Up to this point in the process, all the points have been with respect to the world frame. To calculate the steering angle, the target point must first be converted into the robot frame. The transformation to find the target p_{rf} in the robot frame is simply:

$$p_{\text{rf}} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \cdot [p_{target} - q] \tag{4}$$

The curvature of the arc to intersect the target point with d_w as the length of the wheelbase can be found using a single track bicycle model:

$$\alpha = tan^{-1}(\frac{y_{\text{rf}}}{x_{\text{rf}}}) \tag{5}$$

$$\theta_{curvature} = tan^{-1}(\frac{2d_w sin\alpha}{d_l}) \tag{6}$$

$$\theta_{steer} = \frac{\theta_{curvature}}{\beta} \tag{7}$$

The curvature angle is then down-scaled by $\beta = 1.5$ and then the steering angle is set to θ_{steer} and the robot turns to follow the trajectory.

2.1.5 Velocity Control

The robot's speed is discretized and depends on the position of the robot relative to the start or end of a segment on the trajectory. To maximize speed and thus minimize the time required to complete the track, the robot increases speed on the straightaways but slows through the corners to maintain stability. Simply, if the robot is within a certain $d_{approach}$ of the end of a segment or the beginning of a segment that means that it is turning and needs a lower speed $v_{approach}$. If not, the robot must be on a straightaway and should increase its speed to $v_{straight}$. The distance to the end or beginning of a segment is found using

$D = |q - P|$ where q is the robot's position and P is the end or start point of the segment. Lastly, if the robot was on the last segment and out of the approach zone, the speed was set sufficiently high such that the robot would continue accelerating through the finish line. In the final race, $d_{approach} = 10$m such that:

$$v = \begin{cases} 11.0 \text{ m/s} & \text{if } D \leq d_{approach} \\ 20.5 \text{ m/s} & \text{on straight} \\ 300 \text{ m/s} & \text{on last segment} \end{cases} \tag{8}$$

2.1.6 Custom Trajectory

The final component is the custom-built trajectory. The race trajectory was built such that the car would get as close as possible to the apex of each corner but not carry so much speed that it would slide out on the corner exit and hit the wall. The trajectory was constructed point by point, tuning each vertex until there were no collisions and the car hugged the track limits as desired. The first segment was extended far earlier than the start point to ensure the robot started with speed $v_{straight}$ and similarly the last segment was extended beyond the finish line so the robot would not slow down towards the end.

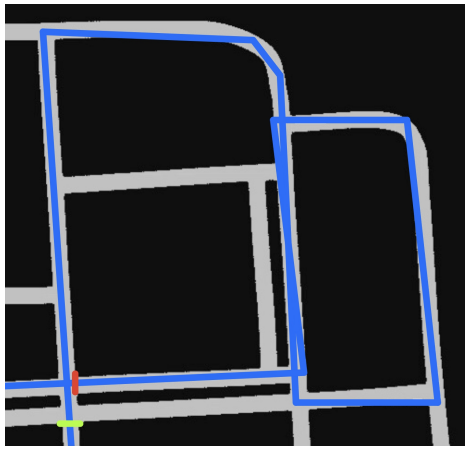

Figure 2: The trajectory used for the final race, note that the vertices at sharp corners allow space for the robot to cut the corner hit the apex of the turn.

2.2 Obstacle Avoidance

Author: Kris Vu

2.2.1 Empty Space Controller

Our strategy for driving our car through the assortment of randomly placed obstacles revolved around control decisions based on LIDAR data received in real time. From the distances received from the LIDAR scan topic, we were able to direct the robot to steer towards windows with a larger amount of empty space, as opposed to towards objects that are already closer. The pipeline for this was to pre-process the LIDAR data, calculate the empty space, tune the steering angle, and steer the robot in that direction.

The goal of pre-processing was to restrict the angles of LIDAR involved in decision making, decrease the sheer size of data, and clip any extreme outliers. Since the LIDAR data goes from a range of angles that also looks to the back left and right of the robot, this extends beyond the range that should be considered when determining where the robot should turn. Instead, we decided to only use the middle fourth of the data, hopefully isolating data values that are predominantly in the front of the robot and not behind. Furthermore, LIDAR data consists of thousands of data points. Even after slicing, there is a large amount of data that is unnecessary in order to get an accurate representation of the world in front of us. In order to improve speed, which is necessary to account for constant perception of boundaries, we down-sampled the data by a factor of 10. This data was then ready to be used in our controller. Finally, we clipped all outliers in the data that were two standard deviations outside of the mean of the data.

Our controller relied on finding the "empty space" in the LIDAR data. We defined "empty space" as the area with the largest recorded LIDAR distance; this meant that obstacles were much further away and the robot could head in that direction for a while without hitting anything. Using a predetermined window size, we incrementally indexed through each set of points in the LIDAR data and calculated the average of all the distances. To ensure we got the most accurate window within the dataset, the windows were allowed to overlap. This, in combination with outlier clipping, worked to diminish large variations in the empty space recognition. The robot could then steer itself towards the middle of the highest average distance window.

2.2.2 Different Approaches

We considered several different methods for the obstacle avoidance task prior to settling on the aforementioned "empty space" controller. Our original strategy was to utilize incremental path-planning in order to build and follow trajectories that moved the robot towards our end goal. The purpose of this strategy was to repetitively call path-planning on short segments in order to account for the dynamic interruption of obstacles. During Lab 6, we implemented a search-based path-planning algorithm, A*, which determines an optimal path to the end goal. However, A* and other search-based algorithms are computationally intensive

and have a longer run-time. This led us to consider RRT (a rapidly-exploring random tree), a sample-based search algorithm, which is much faster. However, we were not able to efficiently and accurately implement the algorithm and the incremental infrastructure in the time that we had. This was a major learning moment for us, as it caused us to make the decision of whether or not to pivot to a neew method.

Additionally, there are several other strategies that could be utilized as well. One of these is a pure-pursuit controller with the added functionality of accounting for obstacles in the trajectory. Another would be exploring vector field histograms, which is a version of our empty-space algorithm, adjusted to account for a more useful form of cost than what we used (distance of obstacles), which can account for both the goal point and interspersed obstacles.

3 Experimental Evaluation

3.1 Final Race

Author: George Chen

Using our pure pursuit controller and hand-designed trajectories, our final race was demonstratively very successful. We achieved a time of 44.8 seconds with zero collisions, making Team 12 the fastest team and the only team to beat the staff benchmark of 45 seconds. Figure 3 showed the actual odometry data against the designed trajectory during the final race. Qualitatively, the robot followed the planned trajectory fairly well. The pure pursuit controller was able to follow the straightaways at specified speed with minimal error. The sharp corners were rounded out to further improve the race performance.
In an effort to evaluate the performance quantitatively, the designed trajectory was upsampled and a nearest-neighbor approach was used to quantify the real-time error of the robot. The error is defined as the Euclidean distance from the odometry data to the upsampled nearest neighbor in the designed trajectory. Mathematically, the average error $\bar{E}$ from a rosbag is defined as,

$$\bar{E} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\left(x_{i,odom} - x_{i,traj}\right)^2 + \left(y_{i,odom} - y_{i,traj}\right)^2\right]} \tag{9}$$

where N is the total number of odometry messages in a recorded rosbag. It is demonstrated that the average error across the entire racetrack was less than 1 meters, sufficient to avoid any collisions with the environment.

3.2 Obstacle Avoidance

Authors: Kris Vu and George Chen

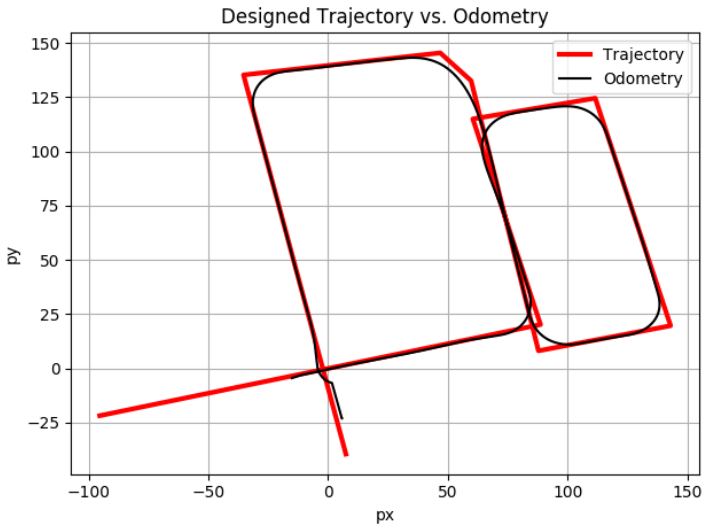

Figure 3: The trajectory used for the final race vs. odometry data. The odometry data visually lines up with the planned trajectory fairly well.

Using the controller that maximizes available space, our obstacle avoidance trial was able to attain a time of 9.85 seconds with 2 recorded collisions. Our robot was somewhat successful, as it was able to correctly identify and adjust to the existence of barriers, but was not tuned to the caliber required to fully avoid them. Admittedly, our performance during preliminary testing ahead of the competition was proven unreliable.

During the final trial, we saw large shifts in where the controller was determining the empty space to be. This led to extreme oscillations. A potential solution to this would be to implement a PID controller, which would hopefully smooth out the rest of the motion.

Throughout the tuning of our parameters, we developed a system to catch errors and ensure we were on the right track. For example, using incremental and conditional print statements, we periodically checked to see if we were breaking the boundaries of road and where the controller was steering the robot. Some of the major issues we were facing had to do with constant veering to one side at the end of the track. These checks allowed us to see that this was due to the two long straightaway at which the end of the track was the vertex. Our robot was seeing the long distance of either of those and driving in that direction.

4 Conclusion

Author: Maria Paula Barbosa

Our final challenge was a successful synthesis of all of the relevant skills and concepts we have learned throughout the semester. Each of the six RSS labs featured important components that aided in the final competition. At the start of our journey as a team, we completed Lab 3 by implementing a wall follower, using TESSE for the first time. Despite our team's tremulous relationship with TESSE, this lab set up a strong foundation to use the simulator. Additionally, we were able to take an initial look at implementing a PD controller and tuning parameters to have a successful run. During Lab 5, we delved into the world of localization. This proved to be an extremely important foundation, as we were now able to locate our robot within its environment using odometry data. Finally, Lab 6 used our implementation of localization to accurately path plan. Accurate path planning coupled with a pure pursuit controller allowed our robot to complete desired tasks in a finite amount of time.

The main objective of the final challenge was to complete two tasks. The first task included following a course trajectory as fast as possible. The second task consisted of completing a shorter trajectory while avoiding obstacles on the course. Our technical approach for the first task included using a pure pursuit controller to follow the custom-built trajectory. This process was similar to our pure pursuit implementation for Lab 6 and included a few modifications to the pure pursuit algorithm so that our robot could accurately follow the custom trajectory without collisions. When working to complete the obstacle avoidance task, we considered a variety of methods including A*, RRT, and using a simple controller method. Due to time constraints leading up to the final challenge, we ended up using a simple controller that prioritizes empty space. After pre-processing the sensor data, the algorithm cycles through until it finds a window between obstacles that the robot can steer towards.

To debug and eventually evaluate our robot's performance in the final challenge, we used both qualitative and quantitative metrics. For the race portion, the pure pursuit was executed and analyzed to see how accurately the robot followed the built trajectory. Using this, the parameters were modified to ensure a faster and clean run every time. For the obstacle avoidance, a similar approach was taken. When testing prior to the competition, our method seemed unreliable. Yet, modifications were made during the final competition that ensured a successful completion of the task.

Overall, our team had a successful final challenge competition. Our team was able to complete the first task in 44.60 seconds without any collisions. This was the fastest time in the virtual competition and ensured us first place. Our second task was completed in 9.85 seconds with 2 collisions. Despite successful completion of both tasks, further iteration could have potentially led to greater

consistency and faster results. All in all, the final challenge was a great wrap-up to the semester.

5 Lessons Learned

5.1 Be Flexible to Changes in Approach (Pau)

During this final challenge, we realized the importance of being flexible to change. When our incremental RRT approach for the Obstacle Avoidance task was not working, we remained stubborn instead of attempting to switch methods. Had we considered switching methods earlier on, we would have had more time to flesh out and tune our empty space method, leading to a more successful outcome in the competition.

5.2 Teamwork Makes the Dream Work (Pau)

I truly believe that the way our team is able to work together and communicate has been instrumental to our success not only in the final challenge, but also every lab leading up to it. I am very proud of what our team has accomplished this semester. Thank you to my team and the RSS staff for a great semester! :)

5.3 Time management is key to success (Rob)

To successfully implement both modules of the final required a lot of time for development and testing. The earlier we start the earlier we know what is not working and can effectively firefight with the help of TA's.

5.4 Small changes have big impacts (Rob)

During the final race, we tuned both the turning speed and straight line speed of the robot. A small increase of 1 m/s in the turning speed and increase of 0.5 m/s in the straight line speed knocked 1.66s off our total time!

5.5 Help everyone do what they like(Kris)

The fun part about the labs this semester was how diverse and detailed they were. There was a lot of different material to work on that dealt with different parts of autonomy. I think it is so important for everyone to feel as though they are contributing in a meaningful way and working on parts of the projects that they wanted to be. Doing so allows everyone to want to contribute beyond just getting things done.

5.6 Big learning starts with small learning (Kris)

When I first started this class, I had no idea what I was in store for. I had very little experience in ROS, autonomous vehicles, controls, and advanced coding. I learned so many different skills that I'm going to be taking with me wherever I go, from working with an (amazing) team on the same code, to debugging skills, to implementing ROS, and even that I really like autonomous vehicles. From the beginning of the class to now, I have learnt so much. It was an absolute pleasure to be in RSS this semester!

5.7 Mustaches are Cool (George)

Throughout the semester, our team dynamics has played a huge role in our success. From the frustrations of debugging weird robot behaviours every lab to seeing our controller pursuit a path so purely, we cooperated together through thick and thin. Instead of *avoiding obstacles* we face it head on to achieve our final success. Mustaches in zoom calls helped too, I guess.

5.8 I Love Robots (George)

This class has exceeded my expectations in many ways. Although having taken more advanced robotics classes before, 6.141 still filled in a lot of my knowledge gaps. I'd imagine it is a great segue to more advanced robotics classes for the underclassmen who would like to delve into this field. Thank you Tanya and all RSS staff for an amazing semester!

5.9 (Lori)

Open communication is important in a team Throughout this semester, our team has become very close, which I couldn't be more thankful for. One major benefit of this was open communication. When a team member was busy or overwhelmed, it was communicated to the rest of the team and incorporated into planning strategy. This allowed us to divert our time and resources efficiently as well as make sure every team member felt heard. Thanks for an awesome semester!

5.10 (Lori)

Don't focus on sunk costs After spending so much time trying to implement and then debug incremental RRT, it was hard to give up and start anew with a much simpler controller. However, focusing on past sunk costs can hinder progress. Instead, prioritizing completing the task and ignoring the pain of scrapping RRT allowed us to implement a working controller for the final challenge.

We would like to thank all of the TAs and RSS staff for all of the time and effort that that was put into this class!