Lab #6 Report: Integrated Path Planning and Control for Autonomous Driving

Team #14

Isabella Do Orozco
Spring Lin
Jace Lu
Kristoff Misquitta
Mark Razanau

RSS

April 27, 2024

1 Introduction (Mark)

Lab 6 addresses path planning and following: enabling our vehicle to determine the most efficient and safe path to navigate a known space given a start point and end point. By building off of previous labs and leveraging localization utilities, this work provides the most sophisticated space navigation methods as of yet for our vehicle. While the robot previously could only respond to local geometric features such as walls and lines, path planning provides a more robust method to automatically navigate between arbitrary points in any space. This was achieved by implementing a Rapidly Exploring Random Tree (RRT) path finding algorithm with a look-ahead pure pursuit navigation model.

When accounting for the computational limitations of our vehicle, we leveraged RRT to quickly determine a sequence of randomized steps that are biased towards the goal pose. As randomization leads to haphazard trajectories, we also added a layer of post-processing to smoothen out the path and reduce potential oscillations as the vehicle travels from the start pose to the goal. As our path is computed as a series of coordinate poses, the vehicle can determine where its location lies with relation to the closest points on the path and follow the sequence of points, with an appropriate look-ahead distance to correctly navigate twists and turns in the path. In combination with our localization functionalities, our vehicle is able to understand where the computed trajectory lies in real space and follow it appropriately.

Inspired by real-world deployments of autonomous vehicles, these methodologies provide a more robust navigation model that is not as reliant on geometric or human-configured features in the environment to function properly. Our specific tasks for this lab were to integrate a path planning algorithm with a pure pursuit method to allow our vehicle to automatically determine and navigate a path in the Stata Center basement. Our work is a step forward in developing an efficient navigation model that will be critical for the final challenge for navigating around the Stata basement in "City Driving" task.

2 Technical Approach

The path following task can be broken up into two main modules: Path Planning and Pure Pursuit.

2.1 Path Planning (Isabella)

Our path was created with a sample based algorithm, Rapidly Exploring Random Trees (RRT). Given an initial pose and end pose, we can map out a non-obstructed path. The algorithm runs with three modules: Sampling and Expansion, Collision Checking, and Node Addition and Termination Check. The produced path is then post-processed to improve optimality.

2.1.1 Algorithm Outline

```
def plan_path():
    start_node = initial pose
    all_nodes = set with start_node
    while counter has not reached set limit:

        # Sampling & Expansion
        if a sample within 0:1 is < 0.4:
            random_pt = goal pose
        else random_pt sampled from map bounds
        next_point = find_next_point(nearest_node, random_pt)

        #Collision Checking
        if map[next_point] is occupied or unknown:
            continue

        #Node Addition & Termination Check
        add Node(next_point) to all_nodes
        if next_point within radius of goal:
            path = get_path from node's parent
            processed_path = postprocess(path)
            publish processed path
```

2.1.2 Sampling and Expansion

The tree starts out with one node at the initial pose. Then, a point will be randomly sampled from anywhere on the map. The closest node in the tree is then found, and a new potential node is created by shifting the original node's coordinates towards the random point by a given step size. In order to improve optimality, there is an added goal bias of 0.4. This means that 40% of the time, the random point sampled will be replaced by the given goal pose.

2.1.3 Collision Checking

The position of the potential node is then compared against the map. If this node lands in either an occupied or unknown spot, then it will be discarded, and the algorithm will return to *Sampling and Expansion*. Because the real robot is wider than only the pixel being calculated, the obstacles in the given map are dilated by 20px to appear larger to the algorithm to ensure the robot will not hit any walls.

2.1.4 Node Addition and Termination Check

The potential node is now added to the existing tree. If this node's coordinates fall within a specified radius of the goal pose, the path from the initial pose is obtained by recursing through each node's parent. Otherwise, the algorithm continues with *Sampling and Expansion*.

2.2 Post-Processing (Kristoff)

While RRT excels at traversing a high-dimensional state space efficiently, it provides no guarantee of optimality. Indeed, the paths generated are often contorted, repetitive, and overshoot the goal. Post-processing offers a chance to clean up the rapidly generated RRT solution, resulting in paths that are impressively streamlined.

The space of post-processors is likely as vast as the space of planning algorithms themselves. Many use some variant of spline fitting or polynomial regression. Given the modest number of nodes in the paths generated by RRT for this problem, a brute force enumeration approach was employed to extract straight line segments from curves.

2.2.1 Algorithm Outline

```
def post_process():
    while start_node is not last of RRT_nodes:
        for end_node from (start to last of RRT_nodes):
            let L = line from start_node to end_node
            if not intersect(L, occupied space on map):
                continue
```

```
        else:
            store (start_node, previous end_node)
            break
    start_node = end_node
    increment end_node
```

The algorithm draws a line from the start node to every subsequent node found by RRT, checking pixel-by-pixel whether that line intersects with pixels on the map that are unknown or known to be occupied. If that happens, the last non-intersecting line is kept, the start node is set to the found end node, and the process repeats, creating straight connections out of the serpentine RRT path (Fig. 1).

Note that this algorithm does not create new nodes; it merely forms shortcuts between existing ones. The upshot is that the number of nodes is drastically reduced, saving memory, but any issues inherent with the path (cutting too close too walls, overshooting the goal) persist.

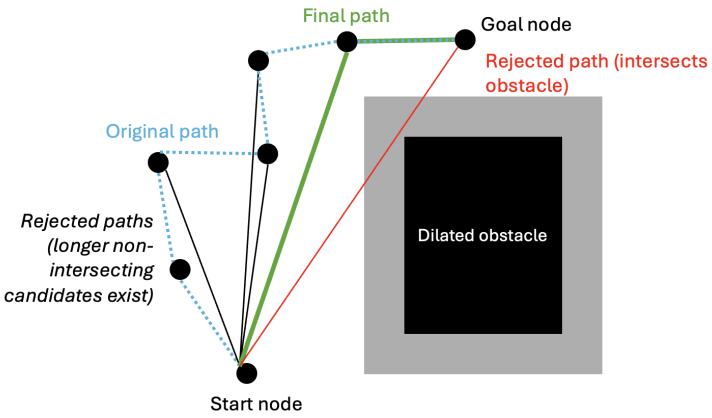

Figure 1: Postprocessing algorithm. A line is drawn to every node from the start node until one intersects an obstacle. The last non-intersecting line replaces the path segment generated by RRT for the start and end nodes considered.

2.2.2 Time Complexity

Forming lines between every pair of nodes and checking their points against a map consisting of more than 10^6 pixels is, in theory, an expensive operation, particularly when those lines can cut across significant swaths of the map. Approaches like dynamic programming (memoization) were considered to avoid duplicated work as lines were redrawn. The most important savings, however, came from leveraging scikit vectorized operations for drawing lines on a 2D array, then comparing them in with one equality operation against the existing map.

2.3 Pure Pursuit (Kristoff)

As opposed to following the standard pure pursuit literature, a "node follower" was used to have the robot follow the generated trajectory. Simply, at initialization, the robot searches for any node on the path that is in front of it (positive abscissa in the robot's frame) and within a reasonable steering angle (meaning, nodes just barely in front of the robot at its extreme right or left are discarded). It then steers by exactly the angular offset of the node from the robot's heading - the robot drives towards the node. Once is within a user-defined radius of the node, it repeats the process, searching for the next node.

This methodology was possible because the post-processor dramatically reduced the number of points in the generated trajectory, effectively having only one at each "kink" in the path. With exclusively straight line segments demarcated by turning points, a node follower proved extremely successful in sim and showed promising results on the real car.

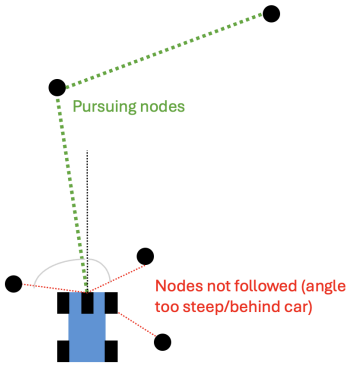

Figure 2: Node follower algorithm. The car considers its angle relative to the nodes in front of it. It does not pursue nodes at too steep at an angle, instead driving naturally towards ones that guide it toward the goal.

3 Experimental Evaluation (Jace)

3.1 Using Goal Biasing for RRT

The RRT path planning algorithm employs random node generation, which inherently leads to variability in the paths computed between the same two points across different executions. This stochastic nature also results in significant fluctuations in the number of iterations required to determine a path. To enhance the consistency of the algorithm, optimize the paths, and reduce the number of iterations needed, we have implemented goal biasing. The outcomes and efficacy of this approach are discussed in detail in the following sections.

3.1.1 Results

To assess the effectiveness of goal biasing, we conducted a comparative analysis of paths generated by the RRT algorithm with and without goal biasing. This evaluation was performed on both straight and curved paths, utilizing the path distances and the number of iterations required as key metrics for comparison. Different rates of goal biasing were also tested to determine their impact on path optimization and algorithm efficiency.

Table 1: Worse Case Straight Line Paths

Goal Biasing	Optimal Path Distance (m) *	Actual Path Distance (m)	Number of Iterations
0.0	29.334	36.046	1640
0.2	28.734	33.032	627
0.4	28.95	31.243	255

*The optimal distance of paths were found using the distance formula for two points: the starting point and the ending point.

Table 2: Worse Case Paths Around a Corner

Goal Biasing	Optimal Path Distance (m) *	Actual Path Distance (m)	Number of Iterations
0.0	17.65	25.912	1319
0.2	17.517	20.504	160
0.4	18.22	21.088	149

*The optimal distance of paths were found using the distance formula for three points: the starting point, the corner point, and the ending point. See Figure 3 for calculation method.

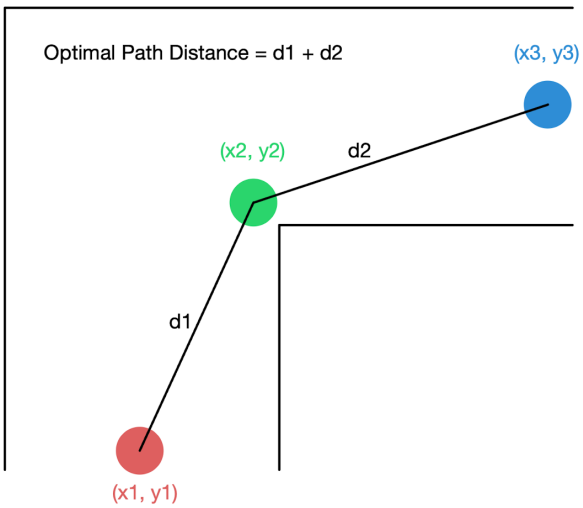

Figure 3: Optimal path distance calculation around a corner.

3.1.2 Anaylsis

As illustrated in Tables 1 and 2, there is a noticeable decrease in the number of iterations required to identify paths as the level of goal biasing is increased. Furthermore, the error table below reveals that the optimality of the paths improves with increased goal biasing. This improvement is evidenced by the diminishing discrepancy between the optimal path distances and the actual path distances, indicating enhanced algorithm performance with higher rates of goal biasing.

Table 3: Errors between optimal paths and actual paths

Goal Biasing	Straight Paths	Paths Around Corners
0.0	22.88%	46.81%
0.2	14.96%	17.05%
0.4	7.92%	15.74%

Equation 1 below was used to calculate the errors.

$$\%error = \left| \frac{actual - optimal}{optimal} \right| \times 100 \tag{1}$$

3.2 Accuracy of Pure Pursuit

To determine if further optimization of our pure pursuit algorithm was necessary, we needed to compare the actual trajectories of the car against the planned

trajectories.

3.2.1 Results

The actual trajectory consists of the XY points outputted by the particle filter, while the planned trajectory comprises the XY points generated by the path planner. Due to the discrepancy in the number of points between these two data sets, it was not possible to calculate the error.

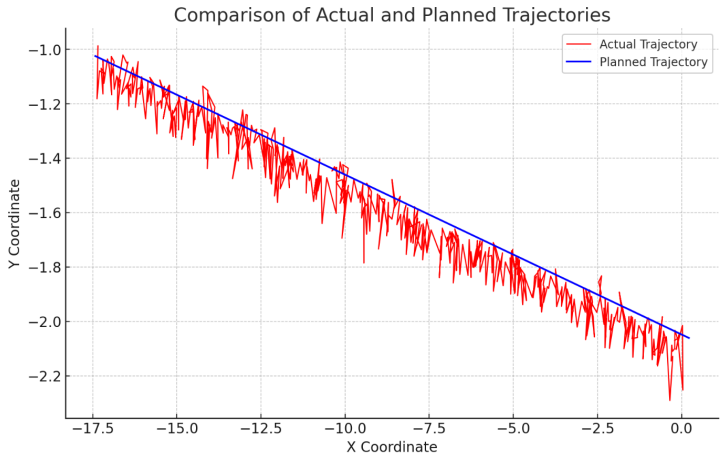

Figure 4: Plot of XY points of the actual and planned trajectories for a straight path.

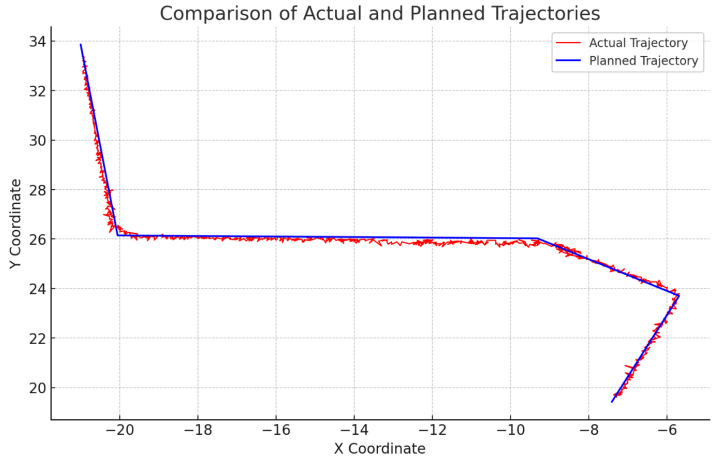

Figure 5: Plot of XY points of the actual and planned trajectories for a curved path.

3.3 Implementation

3.3.1 Path Planning

When the robot was placed in an area of the Stata basement, it utilized RRT to determine a path from the goal point to a specified endpoint. While initially, router issues prevented the use of a scikit package to post-process, future trials did successfully simplify the path in a manner matching simulation. Computation speeds were comparable to simulation, indicating a quick but coarse RRT pass and then a fast post-processing cleanup, as designed.

3.3.2 Path Following

Due to time constraints following hardware issues, the robot was only tested rigorously on rudimentary paths - straight lines and simple curves along a relatively clean, noiseless segment of the Stata basement. Even with the staff localizer, it was suspected that a low update frequency interrupted navigation, resulting in frequent deviations from the planned path as the robot intermittently recomputed a turning angle. The particle filter on Rviz, too, often strayed far from the car's true location, all but eliminating the possibility for recovery. Future work must focus on resolving anomalous behaviors in the localizer and refining the path-following algorithm to reduce undesirable wobble in the car's trajectory. The prevailing belief is that this will only require minor tweaks, since path following (as opposed to planning) relies primarily on components already developed prior to this lab.

3.4 Hardware Challenges

While the simulation developed smoothly, several issues blocked work on the robot. The car was unreachable on the original router; after attempts at debugging during office hours, the router was replaced with one that had no internet connection. A third router was provided in an unsuccessful attempt to resolve this. As the car needed to re-download packages on each start (like ros-foxy-tf-transformations and scikit), large parts of the lab needed to be re-coded to be tested on the car. Once the router hurdles were cleared, teleop consistently threw errors, which were only fixed upon swapping out motor batteries. The sum of these technical challenges was many hours of lost work and a high pressure race to the deadline. The generous help of TAs along the way was greatly appreciated; they enabled us to still benefit from many of the intended learning outcomes of the lab.

4 Conclusion (Mark)

By combining a processed RRT path-finding algorithm with a look-ahead pure pursuit navigation model, our team was able to deploy an autonomous path-planning navigation model that can effectively navigate the Stata Center base-

ment. Our model can be ultimately be broken down into two core modules: the path-finder and the path-follower. By choosing RRT as our path-finding algorithm, we prioritize computational speed over path efficiency when compared to other algorithms such as A*. However with further post-processing of our path, we can remove deviations in the trajectory to smoothen our randomized path to straight segments. This not only allows our vehicle to follow the shortest path between two coordinates on the given map, but also reduces possible oscillations when moving from point to point as they lay along a straight line. For our path-follower, we utilized a point-to-point navigation model with a constant look-ahead circle to consider changes in directionality of upcoming points. As calculated paths can be broken down into a large sequence of poses, our vehicle can easily navigate from pose to pose that intersect with the vehicle's look-ahead radius. This integrated approach allows for a robust navigation model that can efficiently determine and follow the most effective path from a given start point to end point.

Moving forward, we will continue to improve our pure pursuit methods to also consider speed and variations in the path to more closely follow the calculated path. While our current pure pursuit model adequately follows a given path with low error, its accuracy will slightly deteriorate at higher speeds, especially at sudden twists and turns. By integrating a variable look-ahead circle that either decreases in radius when approaching tight turns or increases along straights, we can more efficiently and accurately follow any path as we consider the geometry of the path.

As we look towards the final challenge, path planning will be critical in efficiently navigating the "City Driving" task where our vehicle will be required to navigate the environment while stopping at three specified locations. While also considering variables and obstacles in the environment such as traffic lights, stop signs, and pedestrian crossings, this seems to be an extension of our current path planner and follower. However, rather than focusing on a single end goal pose, our vehicle will have to consider several goal poses in succession. Future work on our navigation model will focus on continuing to improve our localization utilities while also integrating intermediate goal stops along our trajectory with added awareness to different types of obstacles in the vehicle's path.

5 Lessons Learned

5.1 Isabella

I think as this class has progressed, I have gotten more accustomed to unavoidable significant setbacks. It can be particularly frustrating when there is simply nothing you can do to address an issue you are facing, especially the many hardware issues we faced this lab. When this would happen in some of the previous labs, I found myself getting more impatient or frustrated with it. After this lab, it just feels like part of the process now. I really liked how we progressed

before encountering these issues and think it was one of the most solid shows of efficiency in terms of implementation. Furthermore, this week also helped me appreciate my team and us not letting the many problems drag us down or create any kind of frustrations between members. Moving forward, all I can do is hope we do not have as many faultless issues as we did this week; however, I hope we continue to have the same attitude regardless.

5.2 Spring

I was prepared to talk about how much I learned about algorithms and the hardware issues that comes with robotics. However, I instead gained more appreciation for my teammates. I have had a concussion for a while and have been dealing with the following side effects and this week it really showed it's head. Though I tried to still contribute it was made clear by my team that that was not the priority and i needed to focus on getting better. I really appreciate my team for being there and willing to pick up and carry my portion of the work. While I was still able to work on the code, I learned the value of documentation and having a set plan and a set list of steps that I could always fall back on as I worked on pure pursuit. As for my team, I sympathize with their hardware issues as it seems we always have to diagnose and remedy something hardware but I am amazed at what they did given all these roadblocks and hic-ups that came along the way.

5.3 Jace

I appreciated the opportunity to implement the RRT and witness the effects of goal biasing firsthand. However, our progress was repeatedly hindered by various hardware issues. The breakdown of our router led to a two-day delay as we scrambled to find a workaround. Subsequently, we encountered problems with teleoperation and the particle filter. These issues prevented us from testing our RRT and pure pursuit algorithms on the robot until after the briefing. Despite these setbacks, we managed to optimize our simulations significantly, turning the hardware challenges into a valuable learning experience. Overall, this lab proved to be exceptionally challenging, yet the unwavering motivation and problem-solving resilience of our team were truly inspiring. This motivation was particularly evident in Spring's relentless efforts to assist, despite facing medical complications. I am immensely grateful to be part of such a dedicated group.

5.4 Kristoff

I learned the value of strong simulation results to validate my understanding of a theory. Had we relied on results from the robot alone, it would have been impossible to assess the correctness of our algorithm. This lab also encouraged me to think about another challenge of real-life implementations besides noise: extremely large state spaces. When we're finally working on practical systems

in continuous spaces, how we choose to discretize and search greatly impacts the speed and practicality of our solution. Communications-wise, I understood how much a fresh perspective can contribute to an intractable problem, particularly if you ask early, before you get entrenched in one approach. Early flexibility in thinking can alleviate a lot of uncertainty later on.

5.5 Mark

With hands-on experience developing, simulating, and testing our path-planning system on a physical robot, I have a much greater understanding of how these autonomous systems are able to efficiently complex spaces around them. Working on the pure pursuit model especially gave me a much deeper comprehension of how these systems are able to deconstruct complex geometries and tasks into a much simpler sequence of steps: rather than having our robot navigate a path given a start point and end point, it is easier to think of it as having our robot follow a sequence of closely positioned poses that span the beginning point and end point. Communications-wise, this lab definitely tested my patience when it comes to dealing with uncontrollable aspects of a project. Throughout this lab, our team seemed to be getting very unlucky with hardware problem after hardware problems. We had extreme difficulty resolving this issues ourselves due to our lack of knowledge of the system beyond the simulation and software side, and it had us twiddling our thumbs as we waited for staff to diagnose our problems. This often led to a buildup of irritation as it significantly hindered our development process, and ultimately prevented us from being able to complete our testing in time. However, despite these issues, I appreciate how our team was able to not let out our emotions against each other. Based on my experience, in times of frustration, individuals will often let their anger known by letting it out on those closest to them. However, every one of my teammates instead transformed our irritation into fun and humor by accepting that stuff happens. I believe this was also facilitated by a level of trust that we hold between each other that, despite any technical difficulties we tend to experience, we will push through them and find a way to complete our given tasks.