ECE346 Final Project Report

Group 5: Arnav Kumar, Ambri Ma, Brian Mmari, Elizabeth Cotter

Task 1: Successfully completing the obstacle course without manually overriding the robot, hitting obstacles, driving off the track, or crossing the solid lane markings

Method and Rationale:

For our method, we added several nodes to the existing ones from Lab 2. Specifically, we add a node for object detection for the truck and a node for higher-level path planning, before replanning with the trajectory planner. In our higher-level path planning node, we read in the waypoints from the yaml file and keep track of them in the correct order. In addition, we subscribe to the /Obstacles/Static topic in order to read in the static obstacles detected. Then, in this node, we calculate a reference path from the starting point, which is either the original spot of the truck or the current waypoint, to the next waypoint using the racecar routing package, which we then publish for the trajectory planner node. This generates an initial path that is then processed for additional obstacle avoidance to maintain a safe distance and keep the speed reasonable. Then, the trajectory node publishes this path for the robot to follow along. Once the robot reaches the end of the current path, we repeat the planning, adjusting, and publishing process again until we visit all the waypoints in the correct order.

In addition to the overall concept, we made some smaller decisions that had positive impacts. For example, every time we reached a waypoint, we programmed our truck to slow down to give the nodes time to plan the next portion. In addition, we only kept track of the closest obstacles to our truck according to Euclidean distance rather

than keeping track of all obstacles simultaneously.

This overall approach had several benefits. One such benefit was the readability and modularity of the code due to having two nodes, as it was much easier to pinpoint potential errors due to the separation of the path planning, especially since the first node was relatively new, while the trajectory planning node was assumed to be mostly correct. In addition, by using the static obstacles topic, the transition from the simulation to truck was very simple, as we could easily include detected obstacles in our path and only had to stop reading in the phantom obstacles. The small changes we made also had important benefits, as we found that slowing down slightly at each waypoint made our plans more successful in general. In addition, only keeping track of the obstacles closest made plans faster and more efficient in general.

However, there are some potential downsides to these decisions. The usage of two nodes led to sometimes needing to calculate reference paths in both nodes, which could be inefficient and slow. In addition, slowing down at each waypoint may make the overall time slower. Lastly, keeping track of only the closest obstacles worked well in practice due to the fact that there were not many obstacles nearby at any given time. However, this approach may not be the most robust to situations with many obstacles to dodge.

Performance and Challenges:

In terms of performance, our method was quite successful in simulation and on the truck. Unexpectedly, the process of moving from simulation to the truck went very smoothly, only having to change a few parameters and launch files. In addition, the

truck was able to efficiently move from spot to spot, especially when there were few obstacles nearby. However, some weaknesses of the method in practice included the fact that the truck would sometimes go off the lane, leading to confusion of where to go next and resulting in the truck stopping. In general, when the truck was in an unforeseen situation, it had difficulty in finding its way to the next waypoint effectively, leading to several usages of manual control and some obstacle hits. Overall, however, we were pleased with the speed and effectiveness of our Task 1 approach.

Remaining Practical Challenges:

Some practical challenges that remain include improving the robot's behavior around corners, optimizing speed control, and enhancing the robustness of the system to recover from significant deviations from the planned path. Potential solutions could involve fine-tuning parameters like obstacle size and speed limits, incorporating more advanced path-planning algorithms, and implementing recovery mechanisms to handle off-track situations more gracefully.

Task 2b: Safety Filter - Successfully preventing collisions and road departures throughout a manually driven course without causing unnecessary slowdowns, stops, or swerves (outer loop only)

Method and Rationale:

Due to time constraints, we were unable to get a working solution in simulation or in practice for the safety filter. However, here, we detail our planned solution and potential pros and cons. The first step for the safety filter is allowing manual input to the truck. To

do this, we planned to use the Keyboard Control node file supplied by the teaching staff to allow manual movement of the truck in simulation with keyboard input. With this, we would then have to ensure the truck does not depart too far from the road.

For this task, we planned to use a similar waypoint approach as in Task 1, but using only one waypoint and setting it so that the planned path is to simply drive along the outer loop. Then, we could calculate the deviation from the planned path, and when the deviation grows too large, we would override manual input and force the truck back towards the planned path, solving the issue of road departures. Our proposed solution for the obstacle avoidance was similar, as we expected that the planned path would account for the static obstacles in the path. Thus, if manual input attempted to drive towards the obstacle, it would deviate from the plan and be forced again around the obstacle.

Some benefits of this planned approach are its simplicity and potential effectiveness, as we can theoretically solve both problems with the same solution. In addition, it does not require extensive reworking of previous code.

Some drawbacks are the lack of robustness to a variety of scenarios, as it assumes the user simply wants to go around the track. Another limitation is that while the approach should work for lane departures, avoiding static obstacles with this approach may be less effective, as the replanning may result in unnecessary paths and road departures. Lastly, the approach may not scale well to more advanced scenarios, such as more complicated paths or if there are dynamic obstacles, due to its simplicity.

Performance and Challenges:

As noted above, we expect the performance to be effective for road departures, and somewhat effective for avoiding static obstacle collisions. However, the solution may not be robust enough, as perhaps different variations of static obstacles could result in vastly different outcomes, with some potentially being wildly unsuccessful. Thus, we suspect that while the performance may be sufficient for the simple task of going around the outer loop of the track, it may not be sufficient for more advanced paths or scenarios.

Remaining Practical Challenges:

The remaining challenges are implementing the given approach and testing its robustness in simulation. To solve this, we would follow the steps listed above. If we are able to implement the approach, the next challenges would be ensuring consistency in static obstacle avoidance and potentially expanding to paths that are not predefined before running simulation. To solve these, we envision using a similar technique as in Task 1 to plan paths that are effective at moving between points, and then using the approach in Task 2 to make sure the path is followed closely by manual input, replanning and overriding manual input whenever there were severe deviations from the path.

Main Lessons Learned:

From the development and live demo, we learned a lot about the difficulties of autonomous driving and the differences between simulation and the real world. First, we learned that even simple tasks in autonomous planning can require a multitude of interactive components and many hours of debugging. In addition, we learned that even

widely used methods such as ILQR may require lots of hyperparameter tuning and are not particularly robust to all different types of scenarios without modifications. Lastly, we learned that even when simulations work, running simulations on actual hardware may require extra effort to finetune, as there is a delicate balance between software and hardware in the sphere of autonomous driving.