

AUTONOMOUS MAPPING AND NAVIGATION FOR SMALL-SCALE CAR RACING

Kyle Duval*, Will Evers*, Ryan Li*,
Cade Matherly*, Maxwell Miguelino*, Sean Anderson[†]

Department of Electrical and Computer Engineering

University of California, Santa Barbara

Santa Barbara, CA

kduval@ucsb.edu, wevers@ucsb.edu, ryanli588@ucsb.edu

cadematherly@ucsb.edu, maxwellmiguelino@ucsb.edu, seananderson@ucsb.edu

Faculty Advisor:

João Hespanha

ABSTRACT

We consider an autonomous car racing setting with limited information about the racing environment thus requiring real-time localization, mapping, and control of our car when racing against opponents. We demonstrate the use of efficient navigation algorithms for autonomous car racing and obstacle avoidance when limited to onboard sensing and computation. We build off the open-source F1Tenth platform by heavily modifying a one-tenth scale remote control vehicle, thus allowing our Robot Operating System (ROS) to interact with driving controls and onboard sensors simultaneously. Using LiDAR (Light Detection and Ranging), we use a Simultaneous Localization and Mapping (SLAM) algorithm to generate a map of the surroundings in real-time. The vehicle is able to avoid obstacles autonomously and use the generated map to determine its optimal speed. We validate our algorithms in simulation as well as real-world opponent racing.

INTRODUCTION

Autonomous racing involves remote control (RC) cars equipped with sensors and onboard computers in order for the car to detect its surroundings and make driving decisions based on data. Our goal is to build a completely autonomous one-tenth scale racing car and compete against other teams through organizations such as F1Tenth [1], DonkeyCar, or DIYRobocar or by simply contacting other teams outside of these organizations to host our own private race.

The hardware of our car is composed of 3 major components: the chassis, the onboard computer,

*The authors contributed equally to this work.

[†] Graduate student mentor in the Department of Electrical and Computer Engineering.

and the sensors. We use a chassis from a pre-built RC car that contains a motor for propelling the car, an axle for four-wheel drive, and a servo for steering. We use LiDAR as the primary sensor on the car, which gives the distance of objects from the sensor in all 360-degrees. Using the LiDAR data, our control algorithm running on the onboard computer outputs the turning angles that allow the car to race around the track optimally.

The main challenge is to optimize the speed and path of the car. Considering our goal is to maximize speed, optimizing our driving algorithm to maneuver efficiently requires empirical testing for tuning.

RACE SETTING

For the car racing competition, we consider the rules from F1Tenth such that we have the choice of entering our car into one of two different classes: Restricted-Class or Open-Class [2]. The Open-Class competition allows participants to use any hardware component they want with no limitations, meanwhile the Restricted-Class standardizes maximums on your design specifications. We chose the Restricted-Class for our build because the platform has an affordable and accessible platform that is close to our budget. Being in the restricted class also gave us an initial framework for our parts which helped narrow down component selection. If we were to decide to go for Open-Class, the outcome of the competition would likely become more dependent on who has the more expensive parts.

A. Course

The race course under F1Tenth changes between competitions, but there are a few constants. There is always a combination of left and right turns and the course always fits in a space of 10 x 30 meters. The race course materials for autonomous one-tenth scale racing is not exactly standardized for F1Tenth. Historically, F1Tenth races have used two layers of HVAC tube fastened together. This causes some problems as LiDAR sensors can miss scans between the layers of the tubes. Recently, for ease of use, the official track material changed to 12" aluminum HVAC tube. Since our car's main form of sensing is LiDAR, all track boundaries must be tall enough to be detected by the LiDAR. The 12" HVAC tube is tall enough to fulfill this requirement.

HARDWARE

B. Chassis

The main chassis of a F1Tenth competition vehicle is standardized. The Restricted-Class rule set limits the size of the car to one-tenth the size of a F1 vehicle and the power of the drive motor to 3500kV. With this in mind, we chose the Traxxas Slash VXL, which fits within these limitations. The Traxxas Slash VXL is a unique choice for this competition as it has a slightly wider frame than most RC cars, which allows us to more comfortably fit components onto our car. For the purpose of mounting electronics on the car, a secondary chassis needed to be designed.

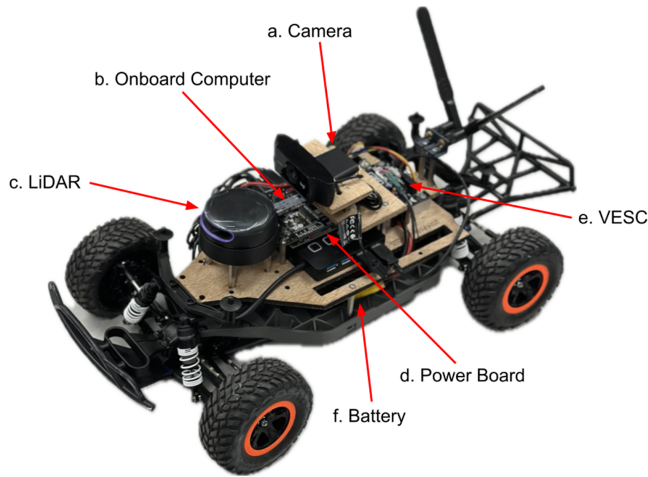


Figure 1: The finished car build with mounted components.

This chassis mounts and organizes the sensors and computing needed for autonomous racing.

We designed the mounting board to fit on our vehicle and laser cut it on quarter inch thick plywood by the University of California Santa Barbara (UCSB) Mechanical Engineering department. While acrylic is recommended by F1Tenth, plywood allows us to more efficiently prototype as we can freely drill holes without risk of cracking the board.

The design itself is based on the mounting holes for the rest of the components. There are long slots cut into each side for the purpose of tucking wires underneath the secondary chassis. These slots are required for protecting loose cables from any moving parts, as well as providing easy access to the components on top of the mounting board. The completed car is shown in Figure 1.

C. LiDAR

The main sensor onboard the vehicle is the LiDAR. The LiDAR sends out a pulsed laser that reflects off objects, which is then read by the LiDAR for a measurement of distance to the object. The LiDAR we use spins the light detection for a proper 360-degree sensing of the location of objects in relation to the car. There are two kinds of detection frequencies for LiDARs: sampling frequency and scan frequency. The sampling frequency is the frequency at which the laser in the LiDAR takes a distance measurement from a specific point. Sampling frequency is less important to us as we will be driving on a track with smooth curves and long straightaways. Due to this, mapping very small details with high sampling frequency is not incredibly important. However, scan frequency is critical as that is the time for the LiDAR to complete a full scan around objects in its 360-degree vision. This is much more important for our application as a faster scan frequency means the car updates its map more frequently. When the car is moving, the car will not sense its location in relation to obstacles around it for brief moments between pulses. To minimize this, we

required a LiDAR with the fastest scan frequency possible that is still within our budget. For this we chose the RPLiDAR-A3, which has a scan frequency of 20Hz.

D. Actuation

We use the VESC6 Mk 5 motor controller, which is a common motor controller used for electric motors that allows precise speed outputs for the vehicle. This is helpful for autonomous driving as it allows us to implement algorithms that use the sensor data to provide very precise instructions to the vehicle.

E. Computation and Networking

We configured the NVIDIA Jetson Orin Nano to run the car. The onboard computer allows all of the other hardware parts to work together as well as running the algorithms for autonomy. For these functions we wanted the card with the fastest computing power that was legal for the Restricted-Class. The faster our car interprets its surroundings, the faster it can make decisions, and ultimately the faster the car can drive.

The Jetson Orin comes in a developer package, which we removed in order to fit the computer onto the mounting board. We replaced the antenna connectors with two 6dBi 2.4GHz 5GHz Antennas from HIGHFINE. These are mounted on the rear of the vehicle on a small 3D printed bracket to ensure strong WiFi connectivity during races.

F. Powerboard

In order to power all the components, we implemented a powerboard. The Traxxas battery is an 11.1V lithium polymer battery, which generally exceeds the voltage of many of the peripheral electronics. Furthermore, a property of lithium polymer batteries is that their actual voltage will continuously drop below their rated voltage as they discharge. While this may not be a problem for the drive motor in the Traxxas Slash VXL, it is for the remainder of the onboard electronics. The powerboard is a PCB that was specifically designed by the F1Tenth organization to fix these problems. Most of the board is a DC-DC converter, which gives a constant voltage to each of the seven outputs. The outputs have two different voltages, with five different 12V outputs and two 8V outputs. The powerboard allows us to run all of the electronics with the Traxxas lithium polymer battery while ensuring a stable voltage output. While most of our electronics are powered directly by the Jetson, this allows us to power 8V and 12V components, allowing us to use a wider range of components. Currently, all of the components onboard the vehicle run off USB power, which is an output from the Jetson Orin Nano. However, the Jetson Orin Nano itself requires a stable 12V from the power board.

SOFTWARE

For our software system, we use ROS (Robot Operating System) which allows all of the hardware components to communicate easily to the on board computer. The reason why we chose to use ROS

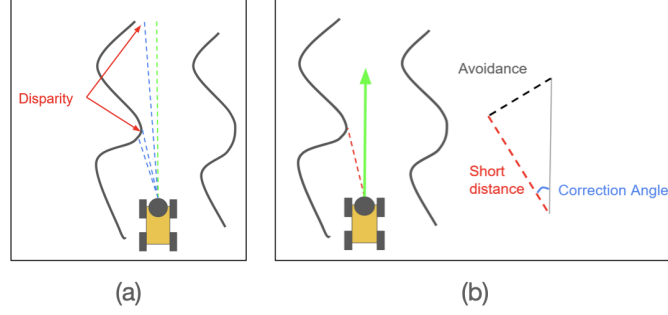


Figure 2: (a) The vehicle detects the largest disparity between adjacent points in a LiDAR scan. (b) Disparity extender turn angle calculation.

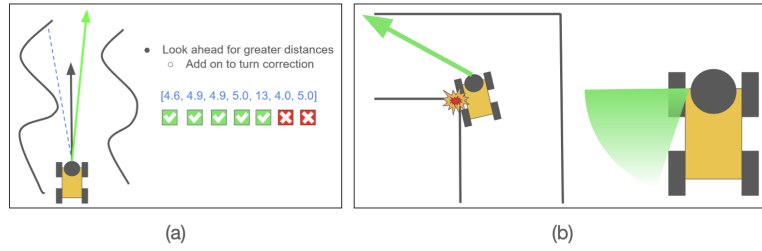


Figure 3: (a) Additional turn angle when longer open distances are detected. (b) Cornering wall avoidance and side wall detection field of view.

specifically is that it is an efficient organizational tool that allows easy communication between different components of the car through information channels called topics. ROS also has a vast library of open-source packages that are integral to making our car function. Our software receives the LiDAR and odometry data and then processes that into meaningful commands for our drive and steering servos so that our car can navigate through a race course appropriately. To accomplish this we use a ROS package called simultaneous localization and mapping (SLAM). SLAM only requires LiDAR and odometry data and uses them to create a map of the vehicle's surroundings. Once we have the map data from SLAM, our car can use the map to calculate the speed at which the car should take turns.

G. Simulator

The F1Tenth Simulator is an open-source simulation in ROS that was created by the F1Tenth community to allow people to test their racing algorithms without the use of any hardware. This simulation helps speed up our algorithm development because we can have multiple people developing and testing algorithms simultaneously without the need of our car and a race track. It also allows us to easily confirm navigation algorithms are functioning, which allows for much easier debugging.

H. Navigation Algorithm

The algorithm we use for navigation uses the LiDAR data and determines the steering direction to properly navigate the track. To do this, we first have to process the LiDAR data. The LiDAR sends out laser pulses that determine the distance of objects around the car in a 360-degree radius. To ensure the vehicle moves in the correct direction, we limit the field of view to 180 degrees in front of the car. The car first searches through the data to find the largest disparity in adjacent data points, as shown in Figure 2(a). The points at which the largest disparity occurs is marked and will be used later in the algorithm [3].

In order for the algorithm to determine a turn direction, the onboard computer averages the scan distances adjacent to the point of disparity on both the right and left sides. The car detects whether objects on the left or the right of the vehicle are closer and calculates a turn direction to navigate away from the closer objects. For example, if the average distance to objects on the left side of the point of disparity is smaller, the car navigates toward the right side of the disparity.

Now that the vehicle understands the point of disparity and which direction to navigate, the vehicle has to determine the angle of steering. This is done with trigonometry as shown in Figure 2(b). The “avoidance” is the amount of distance that the car will avoid the point of disparity. We set the “avoidance” to be a multiple of the wheelbase so the car can comfortably avoid obstacles without any collision. However, setting too large of an avoidance parameter makes the car more prone to oversteering and swerving. The vehicle also uses the smaller distances at the point of disparity to calculate the correction angle.

As an additional fail-safe, the car checks additional LiDAR data points past the angle of correction for any longer distances. If the distances continue to increase, the vehicle increases the turn radius, as a longer straightaway has been detected. This implementation avoids collisions less narrowly and minimizes swerving on straightaways.

The algorithm has some additional features implemented for special cases and ease of testing. In the case of a 90-degree angle (Figure 3(b)), we found that the car’s back wheels would collide with the corner. This was due to the placement of the LiDAR. The LiDAR was placed near the front of the car, and sends a left turn command as soon as the LiDAR clears the corner, without regarding the rear wheels [1]. By using the LiDAR data from the sides and the back of the vehicle, we can tell if there is a wall directly adjacent to the car. When an imminent wall is detected in a certain direction, the car will ignore any turn commands in that direction as to avoid collision. The car will continue to move straight until the rear wheels clear the corner, and the corner is avoided successfully.

For ease of testing the algorithm also uses a reverse function, which allows the car to reorient itself in the case of a collision. This uses the LiDAR data directly ahead of the vehicle and reverses when it detects an object touching the front bumper.

I. Distance-Based Speed Controller

When our car races around the course, it is important for our car to regulate its speed. Instead of going around the course at a constant speed, we implemented a basic speed controller that runs with real-time data. By detecting the distance between the car and objects directly in front of it, this speed controller can determine how fast it increases its speed depending on how far the longest scan distance is received by the LiDAR. If the car sees open space ahead of it, the speed will increase significantly, allowing the car to accelerate through straightaways and slow down on turns. However, the downside to this speed controller is that the car will navigate turns of any severity at a very slow speed to prevent the car from crashing as a safety feature. In order to fix this issue, we have implemented a more advanced speed controller, which the car would have to process after driving its first lap around the course.

J. SLAM-Based Speed Controller

In addition to the basic distance-based speed controller, we also implemented a speed controller that uses a map of our environment. This algorithm is intended to emulate a real F1 driver, who repeatedly practices and memorizes the course before a race. Since our vehicle is unable to practice driving the course beforehand, we use the first lap to generate a map and then switch to the advanced speed controller. The advantage of the advanced speed controller is that the speed of the vehicle can be altered depending on the severity of the turns. The basic speed controller will take all turns at a slow speed, while the map-based controller allows shallower turns to be taken quickly and hairpin turns to be taken slowly.

We generate a full map of the racing course with the use of a ROS package, called SLAM (Simultaneous Localization and Mapping) toolbox which allows us to map out the car's current environment while simultaneously being able to locate itself within the map that it is generating (Figure 4). The SLAM package allows our car to process LiDAR scans of the entire course into a 2D map that can be processed using OpenCV. With OpenCV, we use a thinning algorithm that finds the centerline of the racing course and places reference points along this centerline. Using the location of the vehicle, we find the upcoming three points on the track and calculate a turn angle. The calculated turn angle determines the curvature of the upcoming turn and allows us to optimize our speed depending on the severity of the upcoming turn. This significantly improved our lap time by 16.7% on average. While the algorithm successfully allows the vehicle to maneuver quickly around complex courses, a particular issue arises: the advanced speed controller considers the center of the track, which is not always indicative of the optimal path for the vehicle. The speed controller slows down the vehicle slightly even when a straight path can be taken through the course. This issue may possibly be fixed by a better combination of the basic and advanced speed controller, but more testing would be required.

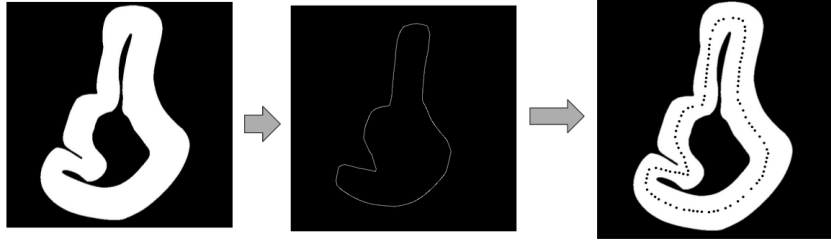


Figure 4: Processing map data using openCV. The thinning algorithm is used to find the centerline.



Figure 5: Test setup for LiDAR skipping on various track materials.

EXPERIMENTAL RESULTS

K. Navigational error analysis

The RPLiDAR-A3 is a middle-range LiDAR that we found is very prone to LiDAR skipping. LiDAR skipping is when a LiDAR laser scan is never returned to the sensor, giving an assumed infinite distance. An infinite distance data point causes problems for the disparity extender algorithm as any infinite values automatically become the highest point of disparity. To mitigate this, we first explored different track materials. To test this setup, we placed the car in a small ring of track material and calculated the running average percentage of missed data points.

From the data in Table 1, we observe that the black reflective surface of the HVAC tube is nearly undetectable by the LiDAR. This can be lowered significantly down to 13.2% by applying a coat of white spray paint. From this data it is clear that dark colored, reflective materials are not easily detected by the LiDAR. In the next test we used a roll of corrugated cardboard to check how the LiDAR is affected by the roughness of the track material. The corrugated cardboard gave an infinity percentage of 21.4% which is a high amount of missed data points. By simply reversing the cardboard so the LiDAR detects the smooth side rather than the corrugated side, the error percentage lowers to 1.1%. The different test conditions appear in Figure 5. From these tests, we now know the optimal desired track materials that give our LiDAR the best odds of success. The ideal track material would be brightly colored, non-reflective, and smooth. This can be achieved by using HVAC tubing with spray paint, as well as using the smooth side of the corrugated cardboard on areas where the vehicle needs better precision.

Testing Material	Black 12" Aluminum HVAC tube	HVAC tube with layer of matte white spray paint	Corrugated Cardboard (rough side)	Corrugated Cardboard (smooth side)
Infinity Percentage	88.2%	13.2%	21.4%	1.1%

Table 1: Infinity percentage results of various track materials

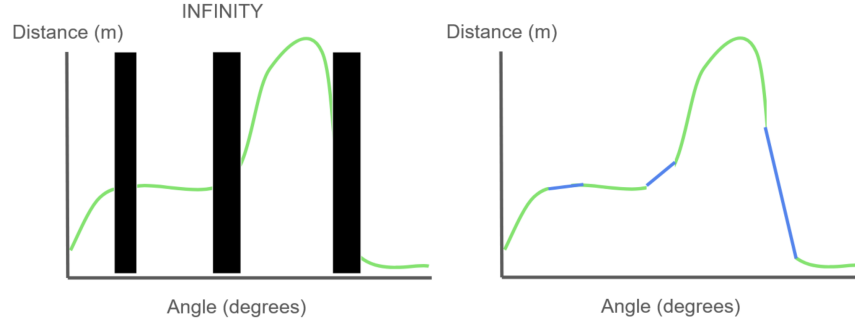


Figure 6: (Left) Example of missed scans due to LiDAR skipping. (Right) Example of linear interpolation between missed scans.

L. LiDAR error mitigation: Infinity Interpolation Algorithm

In order for the disparity extender algorithm to function properly, the usable data needs to overwrite the infinite data points collected from missed scans (Figure 6). While removing the data points would be simple, the location of the infinite values does give us some valuable information. To achieve the best results, we use an algorithm that linearly interpolates missed scans using the adjacent known data points.

This algorithm works well for the intended purpose as the algorithm is able to fill in large sections of missed data points with precision. However, this solution has limits. LiDAR skipping tends to occur in large amounts of consecutive data points, greatly reducing the accuracy of linear interpolation. Therefore, if the LiDAR skips too many data points while important maneuvers need to occur, the vehicle will likely be unable to take the turn. We can ensure the greatest odds of successful navigation by combining this algorithm with track materials that minimize skipping.

M. Racing Results

Mercury is honored to represent UCSB as the first team to race an autonomous car in a friendly competition. Our team was unable to travel to Asia to race in the official F1Tenth races. Instead, we organized an event to be held at University of California San Diego (UCSD) to race our autonomous race cars. By getting into contact with the UCSD Triton AI F1Tenth team, we had the opportunity to race in a competitive environment. This event allowed us to test our car's limits against other autonomous vehicles. Triton AI had a spacious venue that allowed us to build a full-sized 30 by 50-foot track out of HVAC tubes. While our vehicle was capable of head-to-head

races, Triton AI’s navigation algorithms did not allow this [4]. Due to this, we only ran time trials. Through this competition, we received a valuable understanding of how our car would react in a much more spacious and unknown environment.

During this racing competition, we had the opportunity to optimize our car in a full-scale racing environment. By adjusting our speed controller limits, turn angle, and field of vision, we had received better results the more we tested our car on this new track. As a result, we had major advances in reducing our car’s lap time. Along with the success of our navigation algorithm, our vehicle had the potential to reach much higher top speeds than Triton AI’s vehicles. We found that the use of an optimized speed controller is a very important aspect that contributes towards the best lap time for a full-scale track.

Team Name	Key Component	Best Lap Time	Top Speed
UCSB Mercury	LiDAR	12.3s	6m/s
UCSD Triton AI	Depth Camera	21.0s	3m/s

Table 2: Racing Team Results (time trial only)

CONCLUSION

We achieved our goal of creating a completely autonomous racing car capable of navigating through any course. We achieved this goal by using a LiDAR sensor to detect obstacles in the vehicle’s environment along with the disparity extender navigation algorithm. The navigation algorithm works with the speed controllers to ensure the vehicle moves fast and efficiently. Our team sees this project as a research opportunity to provide a proof of concept for the future of autonomous racing in developing driving algorithms.

Our time meeting up and racing with Triton AI at UCSD allowed us to optimize and test our car in a full-size environment as well as get involved in the F1Tenth community. We also got exposed to the wide range of sensors and algorithms that can be used to further improve autonomous driving. We hope that this project is the foundation on which UCSB can develop new solutions for autonomous vehicles and inspire similar works in the future.

REFERENCES

- [1] F1TENTH Foundation, “F1Tenth - learn,” 2023. Accessed 16 Nov. 2023.
- [2] F1TENTH Foundation, “F1Tenth IROS 2021: F1Tenth rules,” 2023. Accessed 16 Nov. 2023.
- [3] “The “disparity extender” algorithm, and F1Tenth,” 2019. Accessed 16 Nov. 2023.
- [4] “Triton AI,” 2023. Accessed 16 Nov. 2023.