CSCI 5302 Advanced Robotics Team Four Technical Report

Alecio Madrid[1], Caleb Escobedo[1], Mary Martin[1], and Scott Scheraga[2]

***Abstract*— In this paper we demonstrate the abilities of a 1/10th scale autonomous vehicle to complete an indoor driving course in clockwise and counter-clockwise directions, recover from collisions, avoid dynamic obstacles, and drift. All computations were completed with an ODroid-XU4 for on-board processing and decision making. Here we outline the decisions that lead to the final product and highlight some of the potential pitfalls of our methods.**

I. INTRODUCTION

Autonomous vehicles have long been researched in academia, and have recently become semi-viable products for automotive companies that hope to move from lane-assisted driving to full autonomy. Due to an increased level of interest, professors at the University of Pennsylvania, Oregon State University, and University of Virginia founded the F1/10th Autonomous Racing Competition in 2016 [1]. F1/10th offers the ability for hobbyists and students to test real-world perception and control techniques for a fraction of the cost and risk associated with autonomous driving [2]. They offer technical diagrams and lesson plans to introduce learners to the key concepts of autonomous driving, but most importantly demonstrate the challenges that are apparent only after assembling a system oneself.

In this paper we outline our approach using a remote-control car, a depth camera, and an on-board computer to identify gaps in car's field of view and direct the car towards the largest gap. Our completed platform is shown in Fig. 1.

II. RELATED WORK

A. Race Outline

Figure 2 shows the final track for the second annual F1/10th race. For our CSCI 5302 Advanced Robotics course, we used a hallway in lieu of a competition track. Both a hallway and independent track offer unique challenges for autonomous vehicles. Fig. 3 shows a recreation of the track we use to test our vehicle. The hallway track has sharp 90-degree turns that completely occlude the vehicle's field of view if it is utilizing a front-facing camera. On the contrary, the real F1/10th course would continuously produce a lack of gaps as vehicles traversing it would make long continuous turns as seen on both ends of Fig. 2.

*This work was supported by The University of Colorado Boulder

[1]Alecio Madrid, Caleb Escobedo and Mary Martin are with the Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA Alecio.Madrid@colorado.edu, Caleb.Escobedo@colorado.edu and Mary.Martin@colorado.edu

[2]Scott Scheraga is with the Department of Mechanical Engineering, University of Colorado Boulder, Boulder, CO 80309, USA Scott.Scheraga@colorado.edu

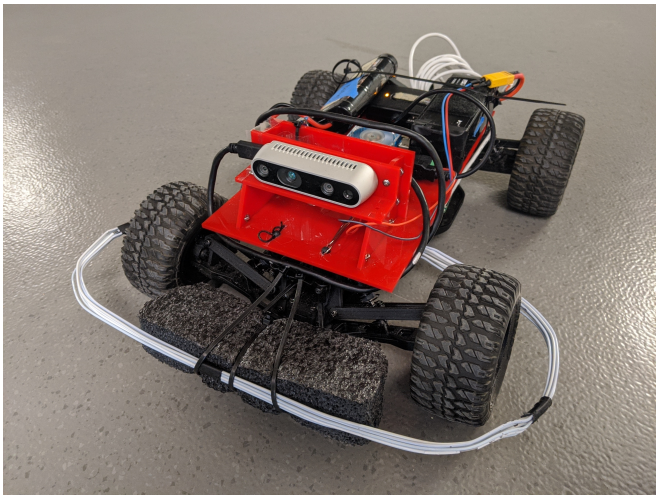

Fig. 1: Photo of completed robotic platform.

Fig. 2: Race track used in the second F1/10th race competition.

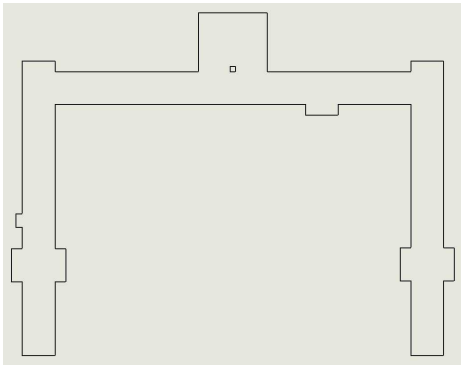

Fig. 3: Map of three-hallway course. Credit:Thomas Lund

B. Planning Methods

We use a simplified version of the "Follow the Gap Method" to segment the scene and plan our trajectory [3]. Follow the gap is known to have potential pitfalls in oscillating trajectory denoted in the improved follow the gap paper [4]. This method has recently shown positive results in the 3rd F1/10th race outlined in a blog post by the competition winners [5]. Other teams have opted to use simpler methods such as taking a square of pixels from the right and left sides of the depth image and then centering the vehicle in between the hallway. While this method may prove to complete the hallway course, it does not guarantee the avoidance of obstacles. If obstacles are placed directly in front of the vehicle it will crash.

III. METHODOLOGY

A. Mechanical Systems

The robot platform is build atop a stock 1/10 AMP MT 2WD Monster Truck RTR ECX03028T2 RC car. No changes were made to the motor, gearbox or steering control linkage systems. For the low-speed midterm evaluation, our robot platform consisted of a single laser cut mounting plate with a velcroed and ziptied components. At the time, a Realsense D435i was mounted low to the ground without a bumper. As the platform would later need to undergo high-speed test runs, a significantly more durable platform for the mounting of sensors and other components would need to be developed. The redesigned platform would have to survive repeated rapid decelerations without any damage to mounted components or significant shifting of the Realsense or IMU (Inertial Measurement Unit) relative to the frame.

The decision was made to rigidly mount the Realsense to to the chassis, as opposed to an adjustable-pitch mount. This was due to concerns that the Realsense could tilt forward during an wall impact event, similar to the head of a crash-test dummy during a fullsize car crash test. The laser cut acrylic upper frame, as seen in Fig. 4 was designed in Autodesk Inventor. The upper frame utilizes the captured-fastener assembly method in order to avoid the need for laborious hole thread-tapping tasks. The Realsense is mounted relatively high in order to center the hallway horizon line in the middle of the RealSense's field of view. The laser cut baseplate is mounted to the RC-car's chassis' existing body mounting posts, and secured with pins. Component mounting positioning is shown in Fig. 5.

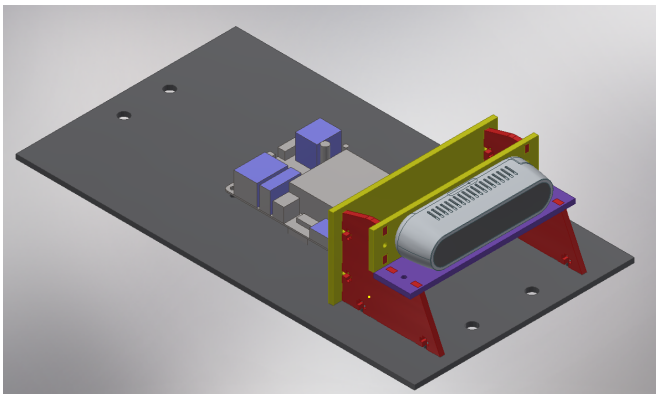

Fig. 4: Isometric view rendering of the robot platform's upper chassis, made of laser-cut acrylic plates. 3D models of the Realsense D435 and ODroid-XU4 are also shown.

The Realsense is attached using both the rear and underside boltholes. The Buck-Boost voltage converter was attached with zipties to the rear of the raised Realsense mount, allowing for easy access the converter's voltage adjustment potentiometer screw. The main mounting plate has screw holes for the ODroid, and enough surface area to securely mount the USB hub and both batteries with velcro. While the batteries were initially mounted far forward, the mounting layout for the final evaluations placed the batteries to the far back end of the lasercut plate. This resulted in a nearly 50/50 front to rear weight distribution. Additionally, the batteries while differently sized, had roughly similar weights. Overall, the weight balance was nearly even left-to-right.

A zip tie is also lightly tightened around the lasercut frame and around both batteries to reduce the likelihood of the relatively heavy batteries laterally flying off of the robot platform during high-speed turns. The Phidget22 1044_O IMU was mounted to the RC-car's chassis using velcro.

The bumper assembly as shown in Fig. 12 was partially inspired by a comment from Jack Kawell, the TA of the CSCI 5302 course, who noted that at least one team in last year's course utilized a coat-hanger-based bumper system. This was deemed by the team to be preferable to the 3D printed bumpers that other team's were considering. This was due to the ability of the coathangers to significantly deform elastically or plastically before material fracture. Additionally, if coathangers were to plastically deform in a collision, the coathangers could be bent back into their intended shape. The Team 4 robot platform uses seven steel coathangers which were bent to provide space for the wheels to turn. The coat hangers were then attached to the RC-car chassis with zip ties.

After a brief discussion with Professor Heckman during Thanksgiving break, the team understood the need to increase

the stiffness of the bumper for front-end collisions. At that time, the bumper would elastically deform rearwards into the front set of wheels during even moderate test impacts. A large block of polyethylene packing foam was deemed to be durable enough for use as the center piece of the bumper. It is attached to both the RC-car's chassis and the set of coat-hangers with a set of zipties.

These design and component mounting decisions resulted in a robotic platform that did not once have a mechanical failure or damaged component due to impact, throughout all testing and evaluation. The bumper performed exceptionally well, and did not appear to take any damage beyond scratches.

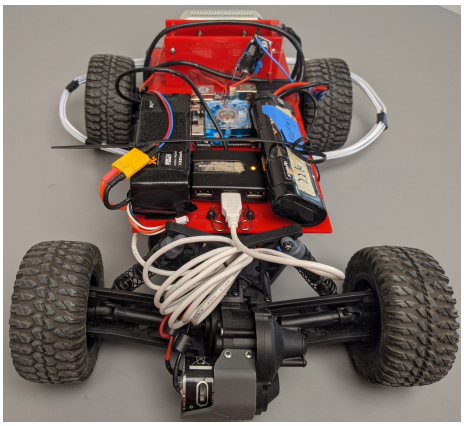

Fig. 5: Photo of mounted batteries, USB Hub, Odroid, voltage converter and Realsense

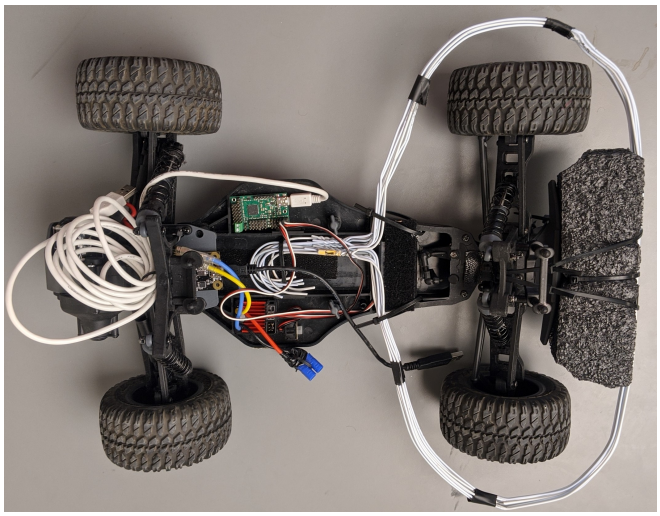

Fig. 6: Photo of the robot platform chassis with mounted bumper components

B. Electrical Systems

The robotic platform's electrical system consists almost entirely of components provided by the CSCI 5302 Advanced Robotics course. The general paths of data and power are shown in Fig. 7. Power is supplied to the ODROID XU-4 from a Hubsan 7000mAh, 11.1V Lithium-Polymer battery, which travels through a Buck-Boost voltage converter, of similar model to the DEVMO DC-DC Boost Buck XL6009. The voltage converter outputs 5V to the ODroid. Our team encountered a large number of low-voltage events which reset the ODroid. Our team was able to narrow down the issue to the voltage converter outputting lower voltages as the Processor Battery used up charge and lowered in output voltage itself. This was resolved by occasionally checking the converter's output voltage, and adjusting its potentiometer screw to return the output voltage to 5-5.15 volts.

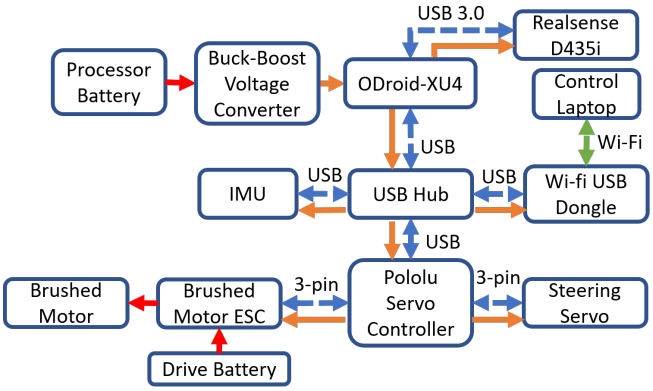

Fig. 7: Flowchart of power and data connections between robot platform components, and connection to control laptop. Red arrows indicate power transmittal above 5V. Orange arrows indicate power transmittal at 5V. Blue arrows indicate data transmittal.

The Realsense is connected to the USB 3.0 port on the ODroid for the higher data transmission rates. The other two USB ports on the ODroid are taken up by USB plugs from an Amazon Basics USB Hub and a Phidget22 IMU. The USB hub is connected to the ODroid-branded Wifi USB dongle and Pololu Mini Maestro 18-Channel USB Servo Controller. The Wi-Fi dongle is used to broadcast a Wi-Fi hotspot that the Control Laptop connects to. The Laptop controls the car over the SSH protocol.

The Pololu Servo Controller receives servo and throttle position data from the Odroid and transmits PWM signals to the Brushed Motor ESC (Electronic Speed Controller).The Servo Controller transmits PWM signals and 5V "logic power" to the servo and ESC through 3-pin cables. The Brushed Drive Motor (540 Size 20T) is powered by a Dynamite 1800mAh, 7.2V NiMH battery. Electrical power for the drive motor is routed through the ESC.

C. Follow-The-Gap Software

We adapted the algorithm outlined in the paper *A novel obstacle avoidance algorithm: "Follow the Gap Method"* in order to traverse the course and avoid static and dynamic obstacles [3]. Initially, we processed the depth image provided by the RealSense D435i by taking a rectangle of pixels as seen in Fig. 8 and grouped objects of similar depth together.

Next in order to have a ”Gap”, where no object exists in the image, we set a threshold to discard objects deeper then a specified depth. For example, Fig. 9 shows the segmented depth image of a hallway, notice that the center of the image, where we want our vehicle to prioritize, is a deeper object than the sides. After discarding objects past the threshold we are left only with objects close enough to be considered a threat that our vehicle must avoid. Fig. 10 shows a visualization of the objects in Fig. 9 after the furthest object, the center of the hallway, has been taken out of the objects list. Next, we calculated the heading angle that directs the car towards the largest gap in its field of view. This augmented follow the gap method works in environments where there is always a gap in the vehicles field of view, but requires extra consideration when there is no object in the field of view.

The hallway used as the course for this project had two 90 degree right or left turns depending on the direction the car was traveling. When the car approached the wall its entire field of view was occluded, leaving no gap for the vehicle to head towards. To remedy this issue we first tried to augment our follow the gap code to head towards the deepest object in the scene. This lead to our vehicle randomly heading towards either the right or left corner of the hallway when we needed to make a 90 degree turn. Ultimately, we settled on a cornering state when the segmentation output no available gap. When the cornering state was initiated, the car would change its heading to -45 or 45 degrees based on the direction we were traversing the course. Along with setting the heading angle to a hard left or right, we decreased the velocity of the car to almost zero, which resulted in the car drifting around corners when completing the course.

Although a large amount of effort was placed in processing and visualizing the follow the gap method using a raw depth image, we opted out of this method a week before the race. We chose to instead convert our depth image to a ROS LaserScan message to use a similar implementation of follow the gap used on a car with a lidar. The underlying logic of the method remained the same, however we chose to process the data similarly to lidar data because we knew it would result in a successful implementation. Fig. 11 shows an example of acceptable trajectories for a lidar scan on a car with a 90 degree field of view.

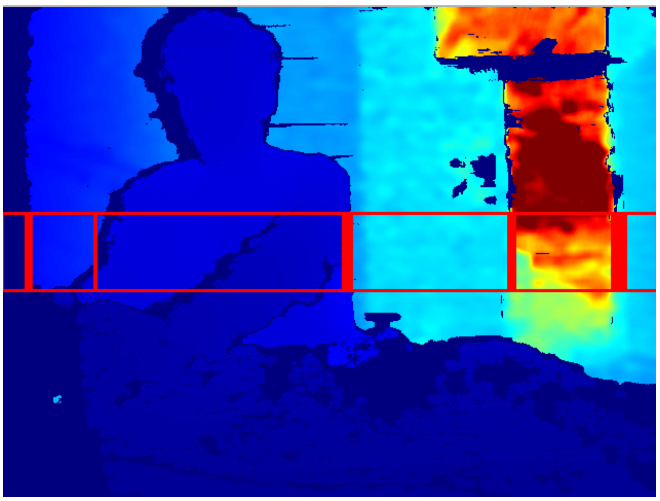

Fig. 8: Processed depth image with bounding boxes grouping objects of similar depth. Blue denotes objects that are close to the camera and red denotes objects in the distance.

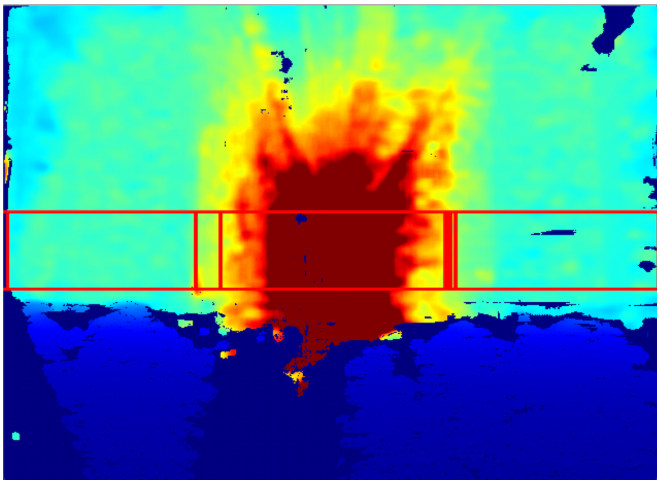

Fig. 9: Segmented depth image of the hallway used as the race course.

IV. RESULTS

Our average of two timed runs for the clockwise and counterclockwise directions were 18.7 and 19.7 seconds respectively. We attempted to complete multiple challenges throughout the process of developing our car. We were successfully able to complete two ”A” category challenges and a ”B” category challenge. These results surpassed the requirements for the course, and can be attributed to a successful execution of the follow the gap algorithm.

A. Ball Avoidance (A-Challenge)

Our follow the gap method allowed us to easily avoid a ball rolling towards the vehicle with one change to the original code, but not to the underlying algorithm. In our code for timed runs, we include a method to navigate when the algorithm finds no gaps. This method effectively allows our car to navigate regardless of sharp turns. However, this caused issues when sharply avoiding dynamic obstacles during testing. After disabling our code for cornering, we were easily able to conduct successful tests with dynamic obstacle avoidance.

B. Collision Recovery (A-Challenge)

We also chose to undertake the collision recovery challenge. We considered two different sensors to accomplish this task, specifically the Intel Realsense and the Phidget22 IMU. After considering potential implementations for both options and researching the Phidget22 Python API, we determined the latter approach would likely be the easiest. Specifically, the Phidget22 API has a getAcceleration() function that

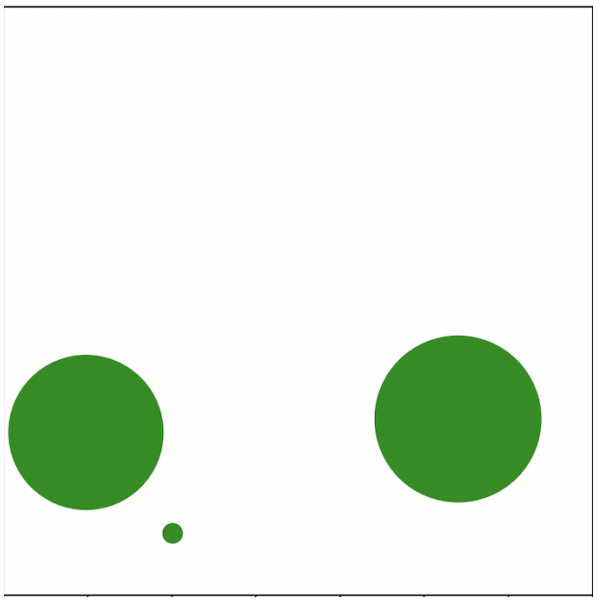

Fig. 10: Visualization of Fig. 9 where the y axis is distance from the camera and the x axis is the cameras field of view from left to right

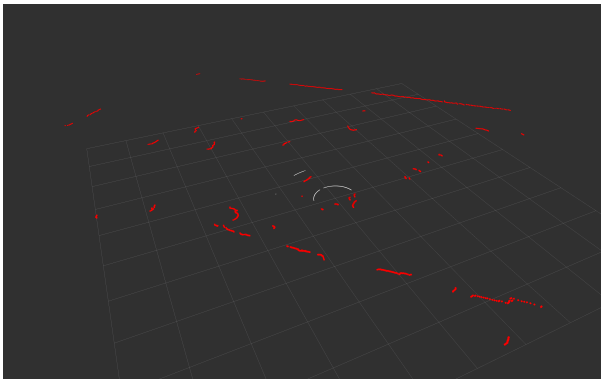

Fig. 11: Photo of a lidar scan where red points are measured form the lidar. Inner white lines represent where no gap exists in front of the car. The outer white arc represents the path the that vehicle will take with no obstacles in the vehicles path.

returns the acceleration values for the x, y, and z directions in gs. Assuming correct alignment of one of these axes with the roll axis of the vehicle, for example x,then x should register a negative acceleration anytime the vehicle comes into contact with an object head on.

We aligned the y axis of the Phidget22 IMU with the roll axis of the vehicle and continuously checked for a large spike in negative acceleration. We found that the IMU measurements were fairly noisy, so we printed out the y acceleration values while the car was not moving to see if we could better characterize the noise. While we did not rigorously characterize this noise, it seemed to be centered around a mean of about 0.06 gs. In order to determine what the y acceleration would look in the event of a head on collision we printed the y acceleration in real time and pushed our car in to the wall at low speeds several times. In each case the value was between -0.2 and -0.25 gs. Without further testing or data analysis we took -0.2 to be a good threshold and set that as our crash tolerance.

In our official run, we started the vehicle in the middle of the hallway and initialized the vehicle in follow the gap mode. At the last second the TA placed a trash can in front of the vehicle close enough that it could not avoid it. Immediately after hitting the trash can, the vehicle detected the spike in negative acceleration and set the motor PWM to neutral (i.e. not moving forward or backward) and straightened the wheels. At this point, the vehicle paused to think about what it had done and consider the dangers of driving recklessly at a high rate of speed. Next, the vehicle backed up at a relatively low velocity (20% of the maximum speed) before coming to a stop. Finally, the vehicle resumes follow the gap and is able to identify the gap between the trash can and wall and navigate around the obstacle.

Fig. 12: Photo of the robot colliding with the trash can (left), backing away from the trash can (center), and avoiding the trash can (right)

C. *Power-slide Turns (B-Challenge)*

While completing our timed runs, our car drifted around each of the corners, successfully accomplishing the drifting task. While driving, the factors that enable the car to drift are the car's speed and the fact that the rear tires lose grip on the driving surface. The conditions of our controls, speed, the weight distribution and the small coefficient friction created the perfect environment for our robot to consistently drift around corners.

D. *Stop Sign (B-Challenge)*

In order to detect a stop sign, our team implemented a functional shape-detecting octagon detector within a Python script using OpenCV and a webcamera, as seen in Fig. 13. Our team decided to investigate a color-independent, shape detection method instead of color blob detection due to comments from the Professor Heckman and Jack Kawell that the stop sign challenge would not necessarily feature a *red* stop sign.

The octagon detector was based on contour approximation, and was built atop the work posted on the PyImageSearch webpage blogpost on OpenCV shaped detection [6] To increase the reliability of the code on the blog post, greyscale and Laplacian gradient filters were added. Additionally, a Convex Hull detector was added to eliminate detected shapes that had convex edges. Our team did not successfully implement a version of the Python script that was compatible with the Realsense within the time-frame of the project. As our team was closer, at the time, to completing other challenges which were more compatible with the team's follow the gap implementation, work towards completing this B-challenge was not continued forward.

Fig. 13: Stop Sign Detector outputs. Top Left: Output of Laplacian Gradient filter. Top Right: Image with applied binary mask. Bottom: Shape detection contours overlaid atop webcam image and dynamic octagon labeling

V. DISCUSSION

A. Track Time Performance and Reliability

The discrepancy in our clockwise and counter-clockwise lap times; 18.7 and 19.7 seconds respectively, can be attributed to our follow the gap tuning. We made our vehicle favor a particular side of the wall in order to allow for more space when making a drift turn. Other parameters that we tuned included turning angle, turning velocity, gap threshold (how far can the vehicle see) and how large a gap must in order to be considered a viable trajectory. These parameters grouped together made each sides' weights completely unique. In the future, we may be able to find a connection between parameters such as turning angle and velocity that allow for turning regardless of specific tuning.

B. Ball Avoidance (A-Challenge)

We specifically chose to implement follow the gap in order to automatically achieve this challenge along the way. This is because follow the gap segments the entire scene into objects that are in the vehicles field of view. We then continuously calculate the angle that will lead us to the largest gap in the scene. This algorithm allows us to find the gap with any arbitrary number of objects in the scene. Also, because the algorithm is consistently checking for objects in the scene it can sense things that are moving and avoid them. One limitation of our implementation is that it cannot avoid obstacles that are moving at rapid speeds relative to its own. However, this is to be expected.

C. Collision Recovery (A-Challenge)

During our testing for the collision recovery challenge, we faced the same issue that was encountered in the ball avoidance challenge. When we removed our code for cornering in situations such as this we were able to complete both challenges. The reason that we faced the same issues in these challenges is because the same logic was used to avoid obstacles in these scenarios. The solution that we implemented works regardless of whether an obstacle is moving in a given scene.

D. Power-slide Turns (B-Challenge)

When we first started to implement turns with our car we made the early decision to tune our car in order to accomplish this challenge. We viewed many other teams instead choosing to slow down their car and take a slight turn when they approached a wall. This method seemed to be less consistent than slowing down to make the turn, as it was prone to slipping at high speeds. Slipping normally caused our car to excessively oversteer and crash into a wall after turning.

VI. CONCLUSIONS

The variety of challenges that we faced allowed us to draw a range of conclusions about the development of autonomous technology. For example, through our execution of the timed runs and "A" challenges with the same follow the gap implementation, it is clear that creating a robust navigation tool allows for us to anticipate many different types of environments and situations.

Something that we did not anticipate was the need to precisely tune the vehicle on different days for successful traversal of the course. With the same parameters, the car tended to perform differently on the course based on factors such as battery power and bearing warmth. There may be additional unseen factors that affected performance from run to run. However, our most effective runs were completed sequentially, on the same day, with nearly-full battery power and after a number of previous runs had taken place to warm up components of the robot platform.

In the future, if we were to continue on our current trajectory, it would be to our benefit to ensure the car is resistant to changes to its environment and hardware. One way that we could create a more robust system is through velocity estimation with methods such as optical flow. This would allow us to ensure that regardless of battery charge, the velocity would remain consistent and have less of an effect on turning.

APPENDIX

A. Potential Impact of Deep Learning On Performance

When developing algorithms for autonomous robots, it is important to consider a variety of methods for successfully navigating a variety of environments. Deep learning can be a valuable tool for overcoming many of the challenges that are faced in autonomous robotics. One tool that could be viable for improving performance is training a neural network on data collected from successful runs using the follow the gap algorithm. This would effectively optimize for different environments and would allow for more efficient processing of incoming data from the RealSense.

ACKNOWLEDGMENT

Thank you to Professor Christopher Heckman, Jack Kawell and University of Colorado Boulder for their support and resources for this project.

REFERENCES

[1] Behl, Madhur. F1/10 Autonomous Racing, http://www.madhurbehl.com/f110.html.

[2] O'Kelly, M., Sukhil, V., Abbas, H., Harkins, J., Kao, C., Pant, Y. V., ... Bertogna, M. (2019). F1/10: An Open-Source Autonomous Cyber-Physical Pl

[3] Sezer, Volkan, and Metin Gokasan. "A novel obstacle avoidance algorithm:"Follow the Gap Method"." Robotics and Autonomous Systems 60.9 (2012): 1123-1134.

[4] Demir, Mustafa, and Volkan Sezer. "Improved Follow the Gap Method for obstacle avoidance." 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM). IEEE, 2017.

[5] Dziubi, Maciek. "Experiences and Thoughts from the 3rd F1/10th Competition." Medium, Acta Schola Automata Polonica, 11 Feb. 2019, https://medium.com/asap-report/experiences-and-thoughts-from-the-3rd-f1-10th-competition-2c46508e2719.

[6] Rosebrock, Adrian."OpenCV shape detection" pyimagesearch, PyImageSearch, 8 Feb. 2016, https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/.