Self-Driving Go-Kart

Shrey Jain, Owen Brake, Veer Khurana

1.0 Abstract

It is clear that the impact of self-driving cars on society is inevitably increasing through major tech companies like Google, Waymo, Uber, and Tesla. The reason for the increased investment into such projects is for the future return on investment of human society. The economic benefits and safety of autonomous vehicles are clear. Travelling will inevitably become easier and more profitable and the increase in safety is the most important.

According to Ericsson, up to 95% of road accidents are caused by driver's mistakes and miscalculations. By applying predictive models to real time data, driverless cars will greatly reduce the impact of human error. Many questions can arise from the idea of self driving vehicles. It is astonishing to note that if all self-driving vehicles were to do was eliminate traffic accidents, that would still be a huge accomplishment. Every single year 37,000 Americans die each year in automotive accidents, and over 1 million automotive fatalities occur worldwide. Self driving vehicles have the opportunity to positivley impact people with disabilities, like the blind, are capable of self-sufficiency, and highly automated vehicles can help them live the life they want.

As outlined, there are endless benefits to the implementation of self-driving vehicles into the real world. Noting that there are some worries, it is important to be able to understand at a fundamental level what exactly is going on with self driving vehicles. The purposes of this project was to enable us as students to be able to understand what is fundamentally happening with a self-driving vehicle on a micro level with a Go-Kart. We would do this using a "first-principles" technique and build the Go-Kart from ground routes up. Through this project we hope to be able to understand and be able to apply elementary machine learning models to a real Go-Kart in the hopes of one day making a conscious impact in the self-driving world later.

2.0 Methodology

Our goal for the project was to build a self-driving Go-Kart. We had to pick where we wanted to focus are time in the beginning stages: a) building the frame of the Go-Kart b) going right to making an already built Go-Kart remote control and then making it self-driving. We realized that we wanted to focus on the technical side of the project versus the mechanical, and thus started to find pre-built Go-Karts that can be modified to become self-driving. As this project was not heavily funded, we had to look for the most cost effective option to purchase a Go-Kart and thus decided to use Kijiji as our medium of purchase.

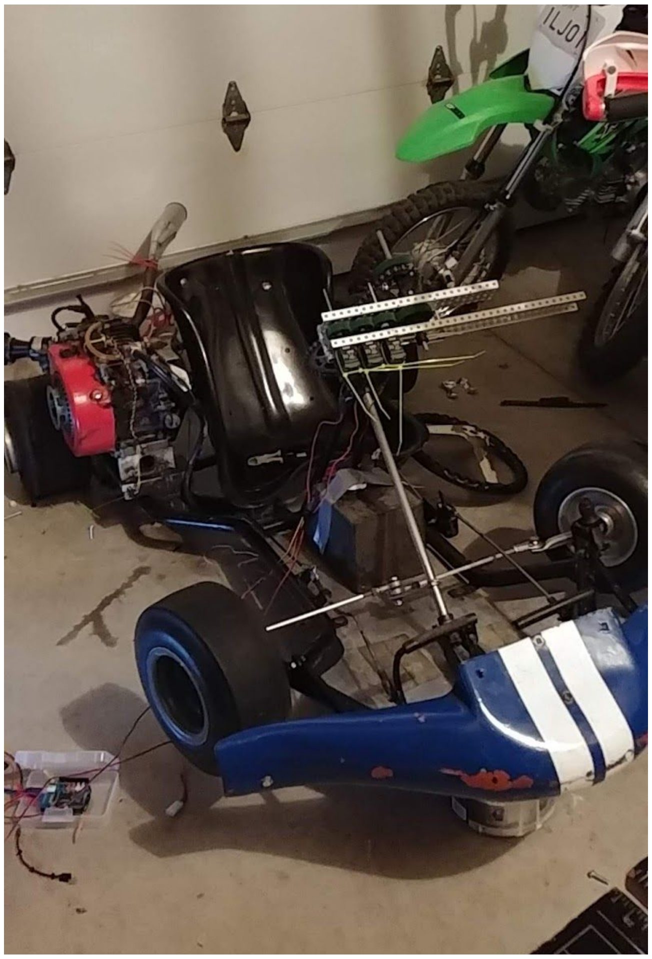

Figure 1. This image shows our Go-Kart while the steering mechanisms are being tested.

We needed to do some testing with the Go-Kart next to see how we could make the Go-Kart remote control and eventually self driving. We first took the Go-Kart for a test drive. At first, the Go-Kart did not look as aesthetically pleasing as one would expect. Some of our immediate concerns when looking at the Go-Kart was how unsafe it looked. The seat of the Go-Kart was tied to the frame by Zip-Ties. Additionally, there was no easy way for us to build a mount for cameras or a computer which was another concern. Our biggest concern however, was how stiff the steering was. If we were eventually going to make the Go-Kart self-driving, we knew that the steering had to be controlled by a motor. However, it was challenging even for ourselves to apply enough force to get the steering wheel to move. So, we had to think small. Our first step to achieving our goal of making the Go-Kart self driving was to have the Go-Kart remote controlled. This required heavy modifications and we had to add a lot of equipment.

2.1 Remote Control Go-Kart

We started this process by purchasing a 2.4Ghz Transmitter/Receiver and an Arduino Motor Shield, which would let us control a motor using the transmitter and receiver. Our first step in making the throttle remote controlled was to mount a servo motor that would pull the throttle cord on the engine. The servo was bought from a local hobby shop and was connected to an Arduino which is connected to a receiver. Our next step was to mount a motor to spin the steering shaft. We knew we would require a strong and high torque motor, so we visited a local electronics shop called A1 Electronics. As we were on a budget, we picked up a used 24V DC motor and a new 24V battery from Canadian Tire. One of the shop employees told us that the motor should easily be able to spin the steering shaft.

Since we knew that the steering shaft would require lots of torque, we decided to have a torque setup, with a small sprocket driving a larger sprocket. This will allow more torque to be generated making it easier for the motor to spin the shaft at the sacrifice of speed. We visited a local bicycle shop and picked up one large sprocket, one small sprocket and bike chain to connect them.

In order to connect the small sprocket to the motor's axle, we visited a welding shop. Once we walked into the welding shop, we were slightly nervous as there were sparks flying everywhere and workers were welding in every corner. We did not have any sort of eye or ear protection. One of the employees told us the cost for the welding job and he started the process. He brought us through the factory and we followed him through the welding procedure. The factory was very loud and due to the vast amount of welding going on, it was difficult to avoid the brightness of the welding. As soon as the sprocket was mounted, we left right away as we were worried for our safety inside that factory.

Our next step was to mount the large sprocket to the steering shaft. This task was difficult as there was no direct mounting spot on the steering shaft. However, there was a flat spot where the steering wheel used to be mounted. We spent many hours drilling through the flat metal in order to create a suitable mounting spot for the sprocket. After the sprocket was mounted, we created a spot for the 24V DC motor to be mounted. Once the motor was mounted, we used a bicycle chain to be put around the set of sprockets which allowed the motor to spin the steering shaft. This task was fairly difficult as we had to cut the chain into the correct size in order for the chain to have some tension and not fall of the set of sprockets.

Once all components were properly secure and mounted, we wired the motor and connected it to the Arduino and the battery. Next, we decided to test and see if the motor was capable of spinning the steering shaft. Right away we realized the employee at A1 Electronics had lied and the motor was very weak as it could not even move the steering shaft a bit. We were very surprised as it was a 24V motor, and it had a torque sprocket setup and was not capable to even spin the steering shaft a little bit. We thought it might be due to an immense amount of friction so we adjusted the positioning of the sprockets/the chain and we applied a lubricant. This did not help one bit, and the motor was still unable to spin the shaft. As we quickly realized the motor was very weak, we began to think of a new method. We used a torque wrench to see how much torque was approximately required and realized a pair of Vex 393 motors could create enough torque.

In order to be cost efficient, we ended up using 3 Vex motors and Vex gears. We created a a torque gearbox. The 3 Vex motors were connected in a 1:1:1 setup which would spin a large gear. The large gear would spin an axle, mounted to that same axle was a small sprocket. The small sprocket was chained to the big sprocket that would then spin the steering shaft. The Vex motors are connected to the Arduino. After all mechanical parts were mounted, the Arduino was programmed. After some testing, we were very happy to find that the 3 Vex motors were performing well and were capable of spinning the steering shaft at a decent speed.

After all parts were in placed, the servo and vex motors were wired properly into the arduino, and the arduino was programmed to communicate with the transmitter and receiver allowing the Go-Kart to be remote controlled.

3.0 Software

When we went to go start developing the software side, we realized that to develop a good system we would have to look at the successes and failures of previous and modern self driving car techniques. Through extensive research we learned that there are two generally accepted methods for lane and object analysis. There is analysis using LIDAR and there is video analysis using high resolution cameras.

LIDAR or Light Detection and Ranging is a process that uses high frequency lasers to create a three dimensional point map of the area. LIDAR is incredibly effective and used by most self-driving car companies however it was infeasible for us since rather then being a large corporation we were 3 Highschool students in a garage. LIDAR has a very steep cost with the lowest tier vehicle-scale sensors going for ~$8,000 USD(velodyne). LIDAR also requires incredibly complex algorithms and 3D analysis which take immense development time and more intense processing power which would be difficult on the minimal frame we were building on.

The alternative to LIDAR was using a more traditional camera, this method uses a high resolution camera and using various image analysis techniques attempts to extract the different objects from each frame in a video. This method is relatively cost-effective requiring approximately ~$50 CAD for each camera used on the device. This method is also relatively more computationally and development friendly as it deals in two dimensional space however both are still a major issue in the development of this software.

Once we had the software methodology determined we began researching and developing prototypes for lane analysis and object analysis using dash-cam footage from youtube. Through prototyping with different methods we determined that using only OpenCV was the best as it was open source, had significant documentation and was free.

3.1 Lane Detection

Lane Detection is a very difficult thing to handle for any form of self driving cars. Modern lanes are never perfect often missing sections, becoming covered in shade and having inconsistent coloring. The best method we found to work was a modified version of the method described in the paper “Autonomous Vehicle and Real Time Road Lanes Detection and Tracking”, and with modification we were able to get a somewhat decent prototype [8].

Before we do analysis of our frame, the first few lines of our lane detector class involve preprocessing of the image.

```
hsled_img = self.filter_img_hsl(frame);
greyscale = cv2.cvtColor(hsled_img,cv2.COLOR_BGR2GRAY)
blur = cv2.medianBlur(greyscale,5)
```

Figure 2. This figure is a snippet of code that transforms the image format.

The code snippet above performs some simple operations to convert the image to a nice format, grayscale the image and then blur it to remove any artifacts providing a more general image for our program to analyze. After the image is processed we do canny edge detection to get the edges of the image, this provides us with the edges of the road as well as the lane lines.

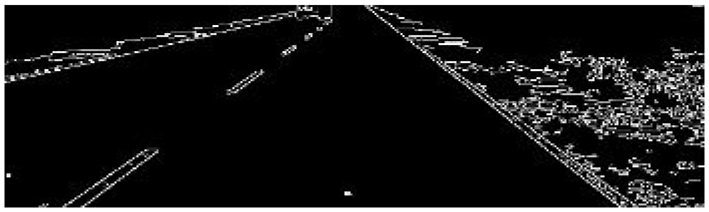

Figure 3. Canneyed image example borrowed from [8]

As can be seen in Figure 3. there are some extraneous edges picked up by the canny edge detector, the edge detector should be tuned to make sure it is only picking up reasonable edges but the majority of this filtering will be dealt with next, when cropping the regions of interest.

After the edges are detected to avoid picking up any artifacts we crop the frame to only pick up the things we know are within our region of interest. The image is cropped to a polygon similar to the one highlighted in Figure 4. where only the road and the lanes are analyzed.

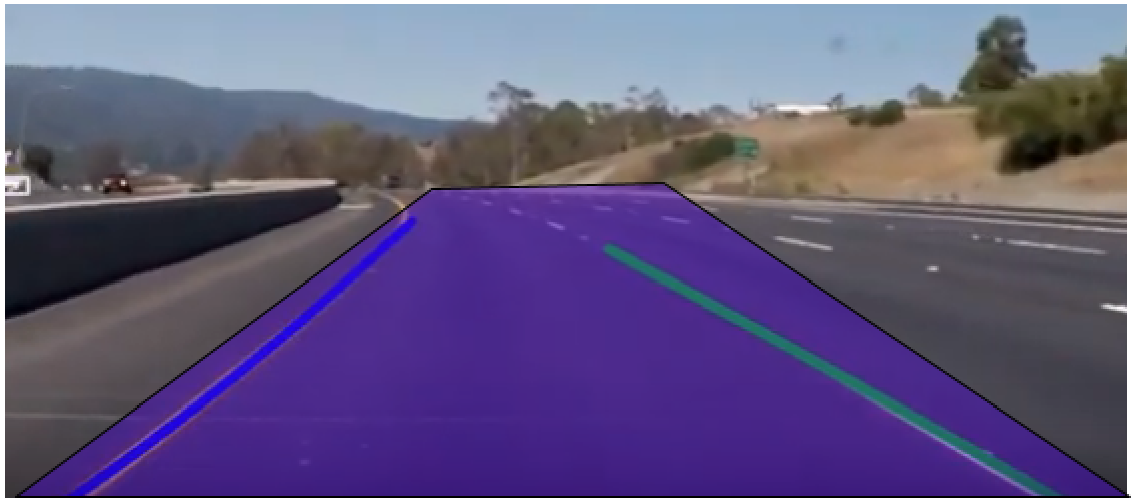

Figure 4. Image of highway with lanes highlighted and region of interest shown.

All this preprocessing is a very important step as the methods of line fitting are only as good as the data they are given. Now that all the preprocessing is done, the complex operations are performed, these operations are all outside the scope of this paper and will not be described. In general a Hough Line Transform is performed to identify straight lines given a set of points. Filtering is done to determine which lines are appropriate and reasonable. Finally the lines are grouped into the left and right of the lane and are drawn onto the image.

3.2 Object Detection

Object detection has always been a tricky thing when dealing with computer vision. There really is no great method to do real time object detection with multiple different types of objects all going at high speed. Through experimentation we realized that OpenCV's Deep Neural Network was the best and simplest method for recognizing and classifying objects in an image.

The code for this object detection is fairly simple as all of the complex operations are dealt with by the OpenCV api. We simply trained our own weights and classes for the neural network by providing test images from various sources and then handed those files to the OpenCV neural network. The neural network returned a set of x and y coordinates which we then drew onto the output as shown in Figure 5.

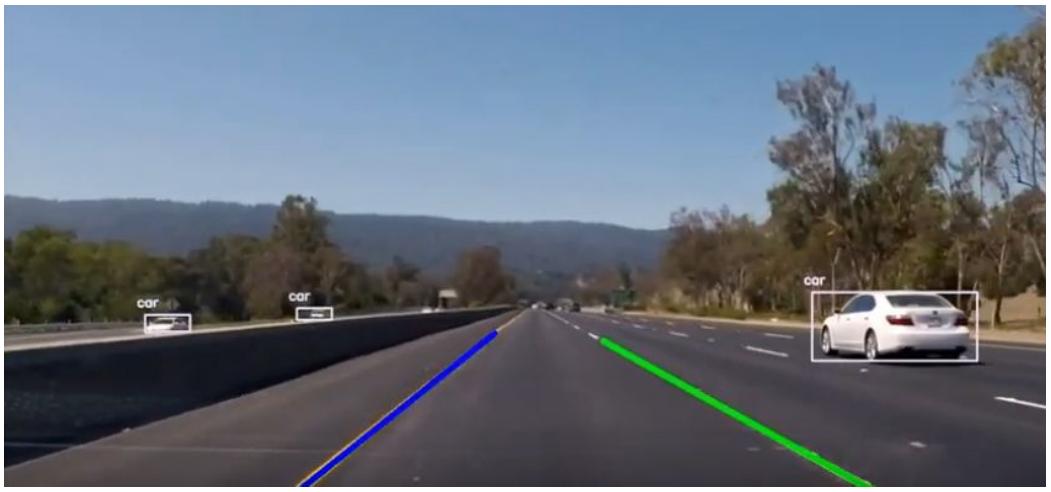

Figure 5. This figure identifies and labels the cars.

3.3 Hardware Interfacing

The hardware interfacing was the final step of the project. As we were never able to successfully implement the self driving software the hardware interfacing was solely composed of the Arduino handling drive by wire. The drive by wire code is fairly simple. The Arduino polls one of the digital input ports and receives a signal from the RC receiver and maps that the range of this signal to the range of some output. The first input is from the RC throttle, the Arduino takes the value for the RC throttle and outputs certain position value for the throttle servo to navigate to. The second input is for the RC steering module, the Arduino takes the value from the RC receiver and regulates the PWM signal to the MC29 motor controllers' signal pin depending on which direction and how much the RC steering module is turned.

4.0 Analysis

Developing this software was a tremendous learning experience for the team. Ultimately we were not able to deploy or further develop our self driving application as the program and in particular the object detection was incredibly processor intensive. The software did not run in real time on any work stations in which we tested and a very powerful computer would have had to be built and mounted onto the go kart to be able to process each frame in real time. This ran outside of our budget and thus we decided to abandon the self driving car capability and leave the software as open source for future developers and engineers to learn and understand from.

Despite the limitations of our software we believe that the output was impressive. We tested on numerous different dash cam footage and it held up fairly well, posted on youtube is the hardest test case our program endured, it involves the car crossing through different light levels, different surface textures and colors as well as dealing with a dotted center line [6]. Overall developing this software was a great experience providing in depth knowledge into different computer vision algorithms as well as providing experience with larger scale software development projects.

5.0 Learning Curve

Throughout the process of building the Go-Kart we hit many obstacles. The first main obstacle that we faced was getting the Go-Kart to turn. The steering of the Go-Kart was very stiff and thus required a high amount of power to turn the axis connected to the wheels. Thus, we needed to calculate the torque necessary to cause the axis to rotate.

What is torque?
Torque is a measure of how much a force acting on an object causes that object to rotate.
How is torque calculated?
Torque is calculated by the formula:

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$$

$$\tau = \|\mathbf{r}\| \, \|\mathbf{F}\| \sin \theta$$

Where:

- $\boldsymbol{\tau}$ is the torque vector and τ is the magnitude of the torque,
- **r** is the position vector (a vector from the origin of the coordinate system defined to the point where the force is applied)
- **F** is the force vector,
- × denotes the cross product , which produces a vector that is perpendicular to both **r** and **F** following the right hand rule
- θ is the angle between the force vector and the lever arm vector.

Torque was not the only new concept that we had to understand and be able to apply. We also had to begin to understand the different types of motors. A DC motor is a motor that uses electrical energy and converts it into mechanical energy by twisting an axel. A servo motor is a DC motor that is coupled with a sensor which allows it to have precise control over angular velocity, position, and acceleration. Most servos are limited to a certain amount of rotation. A stepper motor is a DC motor that makes a full rotation in multiple equal steps. A stepper motor allows for precise positioning and speed. Through understanding these different types of motors we were able to apply the knowledge we had to the Go-Kart.

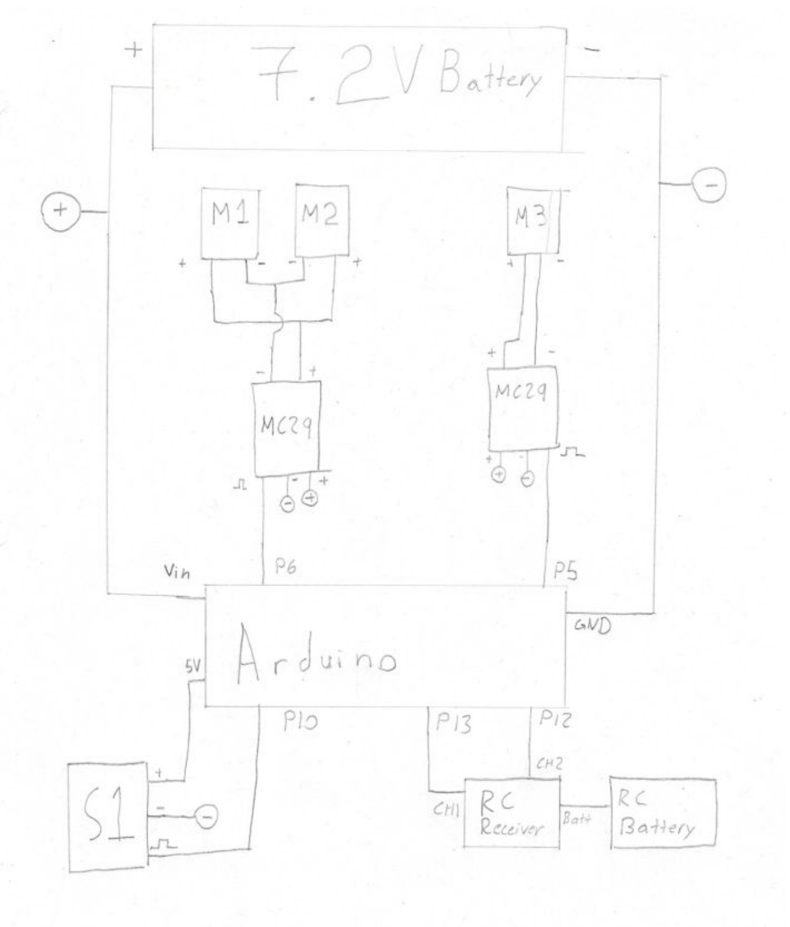

Figure 6. Rough wiring diagram sketch.

6.0 Conclusion

The building of this Go-Kart has been a great learning experience that has given our team an elementary introduction to the autonomous cars domain. We do now realize that the passion that we had for this project is reflective of the passion that we want to invest into future similar projects in the future. We began to realize that anything is truly possible and that even though people around us though it was too premature for us to attempt something so vast, we were able to get it done anyways. Along the way we hit so many different road bumps that caused many headaches, late night work sessions, and ultimately points where we were stuck and did not know where to go. However, we realized that by putting the time in, working collaboratively as a team and sharing a common mission, we were able to get the task at hand completed. We realize now that self learning and applying passion to the real world is inevitably what is going to yield the best results and is shown through this project. We look forward to getting involved in the self-driving car space in the future and take what we learned from this project and apply it later.

7.0 References

[1] https://velodynelidar.com/vlp-16.html

[2]https://www.ericsson.com/en/internet-of-things/trending/targetText=Autonomous%20cars%20and%20the%20future,become%20streamlined%20and%20more%20profitable.&targetText=By%20applying%20predictive%20models%20to,the%20impact%20of%20human%20error.

[3] https://www.youtube.com/watch?v=YJWpPMoWyYc&t=360s

[4] https://www.stat.berkeley.edu/~aldous/Research/Ugrad/Stanley_Yang%20_Thesis.pdf

[5]https://www.forbes.com/sites/davidsilver/2018/09/04/self-driving-cars-will-keep-getting-better-forever/#546f5e8e217d

[6] https://www.youtube.com/watch?v=EaUuswyAPh8

[7] https://github.com/123p10/OpenCVSelfDrivingCar

[8]https://www.gel.usherbrooke.ca/LIV/public/index_htm_files/Autonomous%20Vehicle%20And%20Real%20Time%20Road%20Lanes%20Detection%20and%20Tracking.pdf

[9]https://www.gel.usherbrooke.ca/LIV/public/index_htm_files/Autonomous%20Vehicle%20And%20Real%20Time%20Road%20Lanes%20Detection%20and%20Tracking.pdf