CSCI 5302 - Advanced Robotics: Autonomous Vehicle Challenge

Sai Maddhi
University of Colorado - Boulder
Boulder, CO
sama3612@colorado.edu

Aurangzeb Malik
University of Colorado - Boulder
Boulder, CO
auma9171@colorado.edu

Shreyas Kadekodi
University of Colorado - Boulder
Boulder, CO
shka2478@colorado.edu

Griffin Van Anne
University of Colorado - Boulder
Boulder, CO
grva7970@colorado.edu

Stephen Otto
University of Colorado - Boulder
Boulder, CO
stot7670@colorado.edu

***Abstract*—Creating fully autonomous vehicles is at the forefront of robotics research. In this project several challenges that autonomous vehicles may encounter are examined, with solutions presented for each. The task of performing both localization and mapping is tackled by the creation of an Extended Kalman filter SLAM algorithm. Parallel parking is done by gathering data from GPS and LIDAR and using this to execute a fixed routine which parks the car. Stop sign detection is done through the use of OpenCV. Sparse map creation is implemented using a combination of LIDAR readings and GPS data. The results of each of these implementations was tested in Webots robotic simulator.**

I. Introduction

Self-driving vehicle technology has many profound applications to numerous industries and thus are heavily researched across the world [1]. The challenge of creating a fully autonomous vehicle spans across many sub-disciplines of robotics including computer vision, simultaneous localization and mapping(SLAM), mechatronics, artificial intelligence, and more. Despite the large amount of research dedicated to autonomous vehicles there are many road-blocks in achieving a robust and safe self-driving platform. Three specific challenges that an autonomous vehicle may need to tackle are addressed in this paper: SLAM, parallel parking, stop sign detection. In addition to these three challenges, a sparse map of the environment is also created.

There are many methods that have been developed for SLAM, with more still being created. The method of SLAM implemented in this paper is derived from the Extended Kalman Filter(EKF). This method was chosen because of the vast amount of resources dedicated to its implementation as well as the relatively straight-forward integration required with the vehicle's LIDAR sensor. The EKF algorithm tracks environment landmarks, and updates state belief based of landmark observations. The environment given in the Webots simulation world conveniently has uniform guard rail posts along the length of the track - these were designated to become the target landmarks for the EKF SLAM implementation.

For parallel parking, there are also many different approaches, but our particular method draws upon moving between a few diferent states: aligning, reversing into postion, driving forward. These are done with measurements taken from LIDAR measurements that allow for the precise movement to be done.

Camera based stop sign detection is widely used - the distinctive shape and color of a stop sign make it easily recognizable, and easy to recognize with existing computer vision algorithms.

For this project, we agreed to submit 15 points of projects, but we have submitted 18 points of projects total - 3 extra.

II. Related Work

The Kalman filter was first introduced in 1960 by Rudolf E. Kalman [4]. Since its conception, the Kalman filter has been used widely in many different applications including robotics. Many significant variations of the linear Kalman filter have been developed, most notably the Extended Kalman filter which is extends the Kalman filter's application to non-linear systems.

Simultaneous localization and mapping has been a source of interest for roboticists since the late 1980's [3]. Many different branches of SLAM have emerged since, including Kalman-filter derived algorithms. These approaches assume state uncertainty can be represented by random Gaussian noise and primarily involve a feature-based representation of the environment. Perhaps the most common feature-based algorithm is EKF SLAM, which is implemented in this paper.

Parallel parking is something that was being worked on for a long time now. One prime example was a LIGIIER, and electric vehicle used for the a paper by Igor E. Paromtchik and Christian Laugier back in 1996 [6]. Approaches since the have all followed a stateful design and then application of an algorithm to reverse into a particular parking spot.

III. Methodology

A. Track Maneuvering

1) Controller: Different ways of controlling the car were considered. Proportional controllers were first tested and considered because of the simplicity to get around the track. The team wanted to go for a fast lap time and in order to do that we would have to implement a more complicated controller. A PID controller would allow the car to adjust its speed and steering angle with a variable change.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

The equation above shows the calculation for the error(u(t)) of the system. The steady state error is e(t) and de/dt is the change in error using the previous error from the previous time step. Kp, Ki, and Kd stand for the Gains for each the Proportional, Integral and Derivative terms. Proportional in PID allows the system to be more responsive or less responsive. The Integral term takes into account the magnitude of the error and the duration of the error and by changing this term can directly effect the residual error. Lastly, the Derivative term will adjust the change or slope in error to allow the system to settle to a steady state faster or slower. These can be tuned to fit the system using Ziegler Nichols method.

The car has two different inputs that will directly impact the performance of the car. A PID controller would be implemented separately for the steering and separately for the speed of the car. For the parts of the track with railing the LIDAR measurements will be implemented with 2 PID controllers. When the track has no rails the camera will be used for estimates of the road to control a seperate PID controller.

2) LIDAR Based PID Control: The first thing to consider is what we would perceive as the steady-state error. Steady-state error for a PID controller is the desired final output in comparison to the actual one.

The LIDAR's measurements were used to base how close the car was getting to a collision. The minimum distance from the list of data points taken in the Webots simulator would give us an idea of how close the nearest wall is to our Tesla car and the relative direction of the wall in respect to our car. The minimum distance on the straight paths of the track with the car going directly straight would average to around 3.85 meters. We used this as a benchmark for both our speed and steering.

3) Camera Based PID Control: The camera PID control followed this same general equation, but the overall error function and coefficient values differ. The error function for the steering was modeled by a 3^{rd} degree polynomial finding the error at time t:

$$e(x)_t = \frac{1}{256^2}(x - 256)^3$$

essentially, what is done here is we use the camera to detect the white lane markers, which if the car is correctly aligned it should always be in the center of our field of view. The 3^{rd} degree polynomial allows the steering to be bounded from a - to a + value, and in near center (x values) we produce a low error value, but closer to the edges of our domain [-256, 256] the error value gets more extreme.

In order to collect this x value that represents the lane markers positioning, some computer vision techniques had to be applied. What was done for our approach was once that image is collected (512x256 image), it is cropped for the lower half, darkened, and then we apply a bitwise and operation on a mask vector representing white pixels. From here we have filtered out the unnecessary parts of the image, and now a Gaussian blur is applied, from there the Canny and Hough lines techniques are used in tandem to detect if there are any "line blobs". From there of all the points found (which because the image was cropped should be a relatively small set), are averaged to the get the average x value of the coordinates. This produced value is passed on for use of the error function in the PID controller.

Here is a video of them working in unison to go around the track in roughly 35.9s.

B. SLAM

An EKF algorithm was utilized for the SLAM challenge. This method uses a locally linearized motion and sensor model to estimate the position of environment landmarks as well as the pose of the robot through the use of a Kalman filter.

1) Motion Model: A simple motion model was used to predict the new pose of the robot for each timestep. The equations for this model are shown below:

$$\begin{bmatrix} x(t+\delta t) \\ y(t+\delta t) \\ \theta(t+\delta t) \end{bmatrix} = \begin{bmatrix} -\frac{v_t}{\omega_t} sin(\theta) + \frac{v_t}{\omega_t} sin(\theta + \omega_t \delta t) \\ \frac{v_t}{\omega_t} cos(\theta) - \frac{v_t}{\omega_t} cos(\theta + \omega_t \delta t) \\ \omega_t \delta t \end{bmatrix} + \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix}$$

An inertial measurement unit sensor was added to the car to provide an absolute heading measurement, this measurement is used to estimate the angular rate, ω, over two consecutive time-steps.

2) Measurement Model: The sensor used for observing the environment is a 180° LIDAR. In order to use this sensor in the EKF, we must map the range-bearing measurements to cartesian x, y points. This mapping is done using the equations below:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} rcos(\theta) \\ rsin(\theta) \end{bmatrix}$$

It is worth noting that this measurement is in the local frame of reference of the car - to convert to absolute positional measurements, the current pose of the car is taken into account. All additional EKF equations used for implementation, including Kalman gain computation, were derived from the CSCI 4/5302 lecture slides [5].

3) Landmark Association: The EKF SLAM algorithm stores a catalog of observed landmarks in the state mean. Each time a new landmark is observed, its position is added to the state mean and the state estimate covariance is updated accordingly. Each time a previously observed landmark is re-observed, the correction step of the EKF algorithm will adjust

the state belief to take into account this 'new' information. A challenge in this project was to identify potential landmark candidates, and determine whether or not they had been previously observed. To simply this process, candidate landmarks in this project are limited to the guard rail posts.

First, the raw LIDAR data is grouped into bins according to their proximity to one another. If two points fall within a threshold distance from each other, they are grouped in a bin together. The positions of all points in a bin are averaged to achieve a mean observed landmark position. This mean position is compared to all previously stored landmark estimates and the closest candidate is selected. If the distance between the bin and the candidate is below a threshold, it is deemed to be previously seen landmark observation. This observation is later used in the correction step of the Kalman filter. If the distance is above the threshold, it is considered to be a new landmark, and is added to the environment state.

4) Implementation: The SLAM algorithm is executed at 5Hz which is 1/20th of the frequency of the main controller loop. This rate allows for adequate motion prediction and environment observation, while limiting the frequency of executing the computationally costly EKF steps.

After running a full-loop of the track with only the pure motion model it is evident that the prediction is quite accurate. This observation led to selecting a relatively small additive process noise matrix. This means the algorithm inherently 'trusts' the motion prediction to a high degree. Due to occasionally erroneous landmark observations stemming from the tilting of the car the additive measurement noise is quite large relatively. In addition to making this noise large, the algorithm also ignores LIDAR measurements when the car is executing a 'sharp-turn', which causes the LIDAR angle to tilt dramatically. The threshold for a sharp turn is when two successive car heading measurements have a greater than $5°/s$ difference.

C. Parallel Parking

Parking was done by simply using a GPS, compass and LIDAR sensor. The GPS allowed for localization of the coordinates of the car and the LIDAR allowed for detection of the environment space in the parallel parking world. The controller is made such that the car would move forward using the LIDAR measurements (filtering on from LIDAR angles zero to $\frac{\pi}{2}$). From there be LIDAR or is used in conjunction with the GPS to determine how far certain objects were, find a valid parking stop, stop, reverse into the spot, and perform adjustments. (Note: this a stateful process refer to our Fig. 1 for visual aid)

Now before delving in the specifics of how this was done, here are some assumptions that were made due to the given environment. We had made the assumption that the parallel parking demo required that the car be parked in the space in between 2 cars. We made an arbitrary length requirement for the parking space (P) to be at least 6 m long (in between the front and back car). In addition, the car started at an angle such that it was parallel to the lane marking ($\theta = 0$). If no

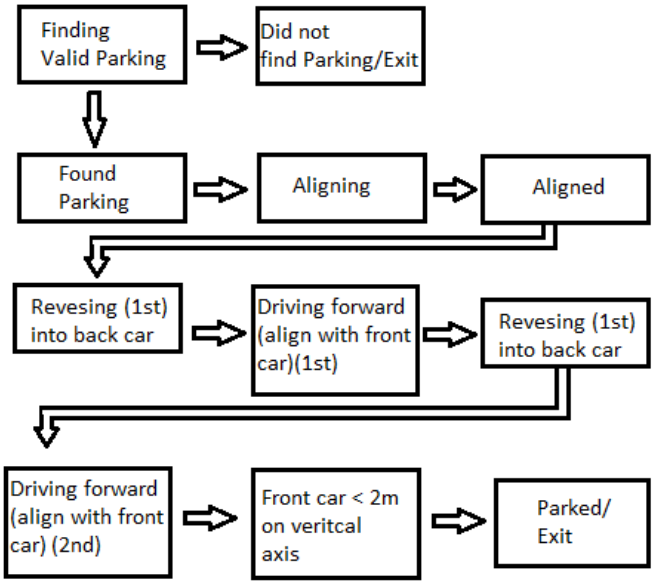

Fig. 1. Figure for describing states and their respective transitions in the parallel parking demo.

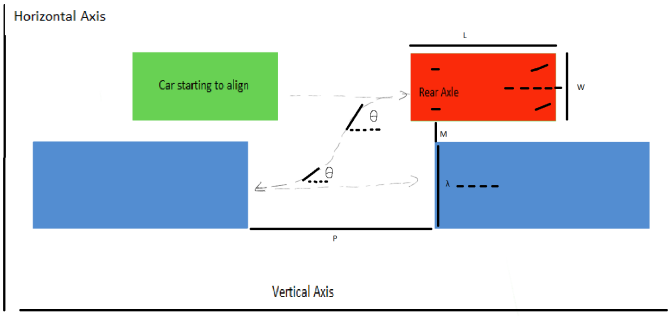

Fig. 2. Figure for describing states and the world for the parallel parking

parking spot of length P or greater was found the car would simply drive to the edge of the world space and then stop. In the case that a parking spot of P or larger length is found, the following will occur. In order to standardize the amount of turning to occur in the future, the car will have been taking measurements since the beginning of the run to determine a rough measurement for M. If M is larger that some threshold ϵ the controller will automatically start adjusting its position on the horizontal position such that the margin, M (distance between the car bodies [refer to Fig. 2]), is minimized so it's within a certain threshold, and it will stop its motion when the rear axle is aligned with the rear of the front car on the vertical axis.

After this was performed, the car will reverse with a positive steering angle, and the steering angle will flip to a negative angle upon reversing on the vertical axis roughly half of length P from the back edge of the front car. The idea behind this movement is sort of a model after the spline path (as seen in

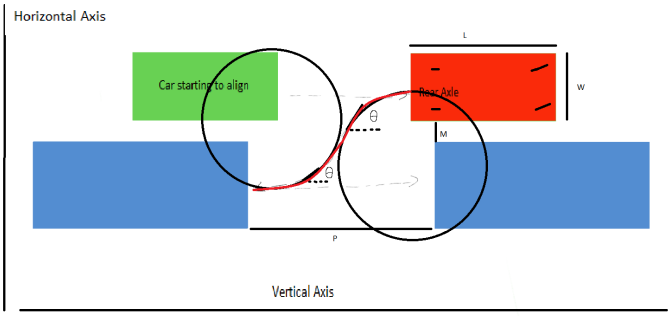

Fig. 3. Figure for describing curving during the initial reversing state

fig 3.) [[4]]. Because we standardized the acceptable value for M, the steering angles from this point did not need to be adjusted to be variable depending on the value for M. While following the path along the spline (drawn in red in fig 3.), when the car has backed up sufficiently within some vertical distance of the back car's hood, we then drive forward. While we are in the state for driving forward (note we are partially already in the parking spot), we use the LIDAR once more to determine if our car is horizontally aligned with the front car (horizontal axis at $\frac{W}{2}$ and $\frac{\lambda}{2}$. This allows us to determine what steering angles to use as we drive just up to the bumper of the front car. We then repeat this process, but in reverse (i.e we enter a second reversing state). Once we have reversed close to the hood of the back car, we drive forward up until there is at least 2 meters of space in front of the car at which point we have successfully parked the car and reach our exit condition

D. Sparse Map

We used the GPS data to localize the car in the environment, and then used the LIDAR data in each iteration to gather information about the surroundings. First we initialize the map at a scale of the real environment. Every iteration we get LIDAR readings, we limit the noise in the system by filtering out readings that are outside of the maximum angles that we define dynamically based on the environment we are mapping. Choosing only readings that are close to the center of the car allows for mapping areas that are relevant to which direction the car is facing. This is important because it enables us to direct the car towards the sections of the map that we need to focus on mapping.

Even with the precaution of only taking readings from the center of the LIDAR sensor, there is frequently measurements that are noisy which can dilute the accuracy of the sparse map. We combat this noise by incrementing the value of a pixel that maps every time we get a reading. As opposed to drawing a reading directly on the map based on the raw readings, we only depict a pixel on the map if its value exceeds a preset threshold. With this stipulation, the display map will only reflect values that are repeatedly seen by the LIDAR. The repetition ensures that values on the map are not noisy and in fact reflect the true state of the environment.

Another aspect to note is in regards to how we use the GPS values in conjunction with the LIDAR readings. Initially, every LIDAR reading we get has an angle and distance measurement that is in the car pose instead of the world pose. We normalize this data and bring it into the world pose by using the $r\cos(\theta)$ and $r\sin(\theta)$ equations from part B. This conversion allows us to map the (x, y) point straight onto the scaled display map.

E. Stop Sign

The stop sign detection was implemented using the OpenCV library in Python. Initially, an XML training set was acquired online, which contains features relating to stop signs. From there, the data is passed into OpenCV's cascade classifier. Object Detection using Haar feature-based cascade was first proposed in 2002 [2]. OpenCV implements this method in order to allow for feature based object recognition. The image from the camera is saved and then passed into the classifer, which returns whether it detects a stop sign or not.

IV. Results

A. SLAM

The figure below shows landmark observations and the path of the car using dead-reckoning from the motion model only:

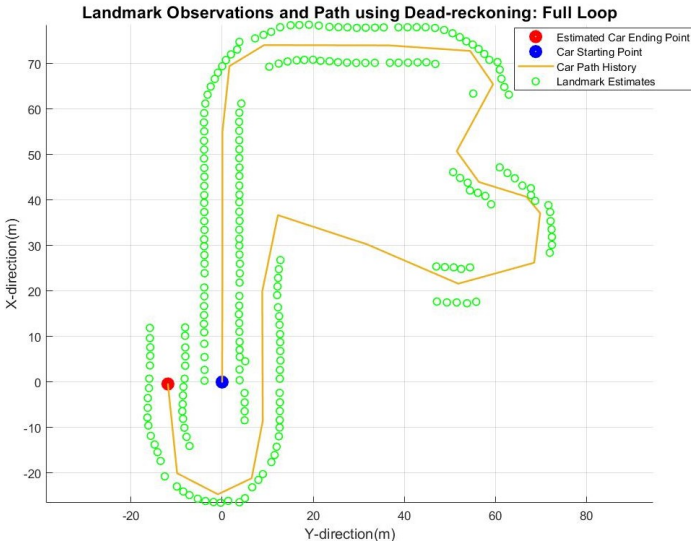

Fig. 4. Mapping of environment using purely motion model prediction

As mentioned previously, this appears to give a quite good estimation of the landmark positions in general without the Kalman filter correction step. The final y estimated position does end up offset from the true position, however the x estimated position is very accurate. The offset y value is likely due to drift accumulated throughout the path which does not get corrected by repeated landmark observations.

Shown below is the result from the fully implemented EKF SLAM algorithm for an entire loop of the track:

The total track completion time with the full SLAM implementation is 1 min. and 26 sec. This slow-down in track completion time was done intentionally to limit the effects

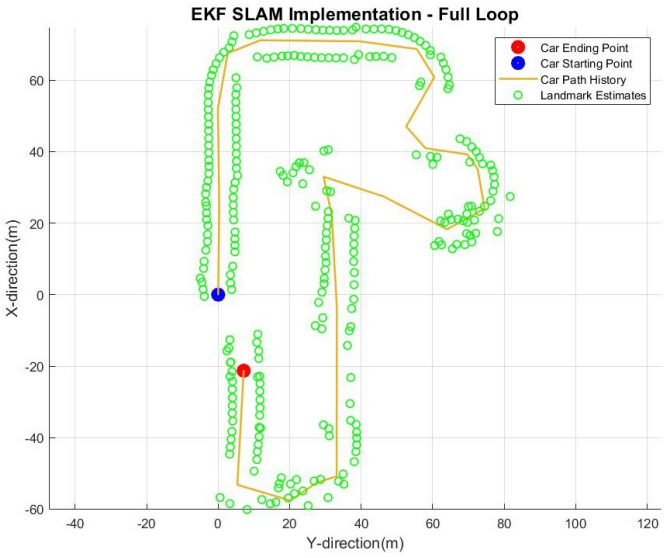

Fig. 5. Mapping of environment using EKF SLAM

of sharp turning and acceleration on the LIDAR tilt angle. With the inclusion of the state belief update step of the EKF algorithm, the estimated car path and landmark states were subject to some inconsistent results. At the beginning of the loop the algorithm seems to do an adequate job of estimation, with the path and landmarks being consistent with the given track. Near the third significant turn there appears to be a build up of erroneous landmark estimates. When a landmark is incorrectly observed, the difference between the expected and actual observation can be quite large. This leads to a bad state belief update which has downstream effects for all subsequent EKF iterations, leading to the inaccurate final car position estimate.

The figure below shows the landmark and pose estimate before and after an EKF correction step:

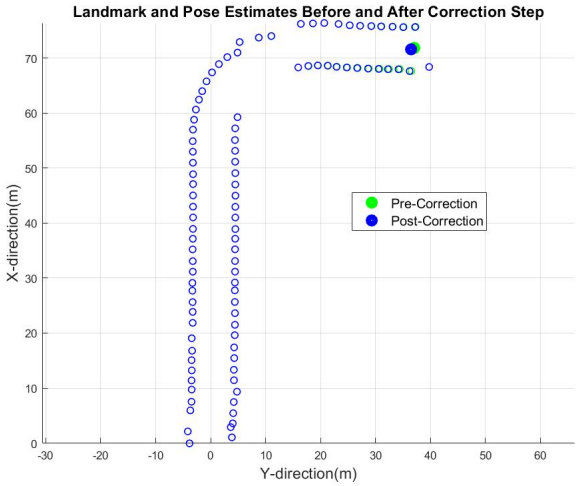

Fig. 6. Landmark and Pose Estimates, pre and post correction

This figure shows that the EKF implemented is indeed updating its state belief based on additional environment data provided from a LIDAR landmark observation. The correction made is quite small, likely due in part to the small additive measurement noise(trustworthy sensor), and the small difference between the expected and actual observations.

B. Parallel Parking

Two timelapse videos were recorded to depict the variance that our parallel parking can handle. In the first video, we place the car at a nearly $7m+$ offset from the original position, and the car successfully manages to park it self in between the car and the red van. In the second video, the car was shifted about $2m$ to the right of the original position. Additional testing was done, and it was found the van could move about $2m$ downward, but any more and the controller as it is will not be able to properly fit the car into the space. Conversely, due to the threshold set, the red van can be at most $2-4m$ further (vertically) from the car. After that the threshold for P is far too big and it's not considered a valid parking spot, but this can be adjusted for if we consider a different threshold. If no spot's meet these requirements the car will drive to the end of the world and stop.

C. Sparse Map

The sparse map code was modularized and placed in a class which allowed for us to use the mapping functionality in multiple worlds. The results of the mapping can be seen in both the SLAM challenge track world as well as the parallel parking world, both displayed below.

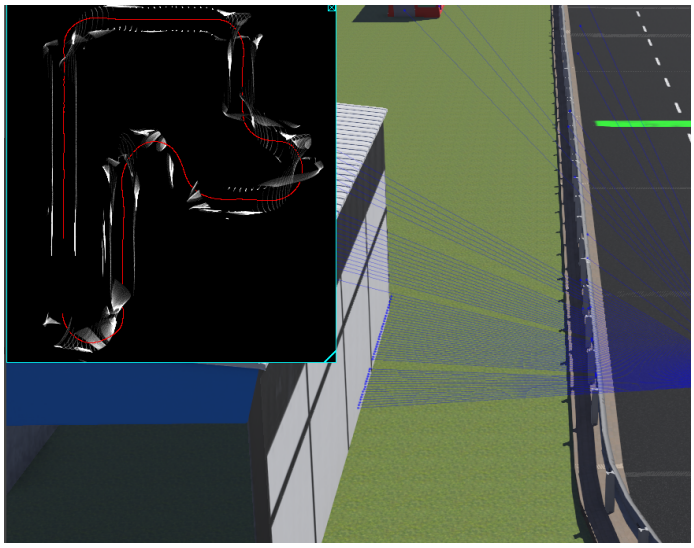

Fig. 7. Sparse map generation while going around the track.

The above map is a depiction of the parallel parking world that was produced by the path the car takes when parking itself. The sides of the parked cars are visible which are in range of the LIDAR sensors that are not filtered out.

The map depicted above shows the LIDAR sensed when navigating around the track autonomously. There is noise near

the turns because of the tilt of the vehicle that results when taking a sharp corner or bend. This causes the LIDAR to sense the raised sidewalk and asphalt as well. This can be seen by the corners of the track where there is unusually noisy data and much more of the track is highlighted compared to the usual rails on either side that typically are the only measurements that are leftover after the filtering.

D. Stop Sign

The stop sign detection works in the overwhelming majority of scenarios in the traffic sign world. Provided that the stop sign follows the assumptions (stop sign on the right, camera of certain resolution, stop sign of a minimum size), the car slows to a full stop with a reasonable margin and then starts again once it has reached a complete stop. The result was captured as a time lapse here.

V. Discussion

A. SLAM

The EKF SLAM results were mixed. On one hand the algorithm showed good performance at grouping measurements into landmark observations and the ability to update state belief based on these observations. On the other hand, when the algorithm encountered erroneous landmark observations the state belief was quickly derailed from the 'true' state and the algorithm was not necessarily able to fully recover.

The motion model used was shown to be overall very accurate and was able to produce as good, if not better mapping of the environment on its own compared to that of the EKF. If more rigorous landmark detection criteria were used, such as requiring multiple candidate observations before adding as a landmark, then the results of the EKF would likely improve dramatically. The algorithm could have also been made more robust by allowing for any LIDAR detected object to become a landmark, not just the railing posts. This would mean the performance would become less sensitive to observing unexpected landmarks.

Another method which could have been implemented to improve the SLAM results is a loop-closure correction. If the car recognizes an area of the map where it has been before, for example when it crosses the starting line after completing a full loop, the state belief can be corrected accordingly which will also collapse the uncertainties in the covariance matrix. Repeating this process multiple times would likely result in better and better mapping. Since the rail-post landmarks chosen are all uniform it would be difficult to identify points which have been visited before based purely on landmark observations. For this reason the onboard camera could be utilized to help identify loop closure points.

An entirely different, non-feature based SLAM method such as Grid-based SLAM could have also been chosen in order to bypass these landmark identification challenges - perhaps leading to better results.

B. Parallel Parking

The parallel parking results were mixed. The major ability of it is to handle wide horizontal changes to its initial environment (starting super left or to the right), and still land in the right category. However, there is a challenge of arriving in the right parking if the vertical distance P between the van and the car is too big or not sufficiently big enough. The current approach is a bit picky on this particular matter.

We could improve this by instead of using a piece wise function to model the spline trajectory in the initial reversing stage, we could utilize more advanced sinusoidal functions that would allow us to much more accurately slip into the parking slot far better than how it is done now [6].

C. Sparse Map

The sparse map outputs are close to the results that we expected. The noise that we saw in the SLAM challenge world was initially concerning because the error seemed to be in the mapping code. After further analysis, we realized the LIDAR readings were tilted when taking a hard corner. This showed that the error was resulting from the tipping car which sensed the ground which came into range when the car tilted. The raised sidewalk was also coming into the LIDAR readings even after our filtering code. To fix this in future experiments, one proposed idea is to only accept LIDAR distance measurements that are within a range. We already filter out data that is too far away which would be more noisy than data closeby. Likewise, filtering out data that is too close to the car may be a good workaround to avoid picking up ground and raised sidewalk measurements. However, this may not always work because the ground readings when the car tilts are not entirely limited to the first meter or so in front of the car. Furthermore, if we were to instantiate a range in which the LIDAR readings would be accurate, we would be filtering out data that may be pertinent in other aspects – like the SLAM code needs to have access to readings that are very close to the car since it is a safety hazard if the car cannot see what is directly in front of it.

D. Stop Sign

The stop sign detection worked largely as planned. In order to emulate real conditions, a filter was applied to restrict detection to the right hand side of the camera's view. However, this means that should a stop sign be placed in the center of the field of view, the detection will fail. Additionally, since the stop sign is shape based, a similar shape in the background may cause the car to stop, and as built the system will not fare well with multiple stop signs in a row. Otherwise, however, the system works as expected.

VI. Conclusion

Acknowledgment

References

[1] Heineke, Kersten, et al. "Self-Driving Car Technology: When Will the Robots Hit the Road?" McKinsey and Company, McKinsey and Company, 26 Feb. 2019

[2] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In Image Processing. 2002. Proceedings. 2002 International Conference on, volume 1, pages I–900. IEEE, 2002.
[3] Durrant-Whyte, Hugh, and Tim Bailey. “Simultaneous Localization and Mapping: Part 1.” IEEE Robotics and Automation Magazine, June 2006.
[4] Urrea, Claudio, and Rayko Agramonte. “Kalman Filter: Historical Overview and Review of Its Use in Robotics 60 Years after Its Creation.” Journal of Sensors, vol. 2021, 3 Sept. 2021.
[5] Hayes, Brad. ”EKF-SLAM 1-3” [PowerPoint slides]. CSCI 4/5302 Course Page. https://canvas.colorado.edu/courses/77519/files, 2021
[6] Igor E. Paromtchik, Christian Laugier. Motion Generation and Control for Parking an Autonomous Vehicle, Proc. of the IEEE Robotics and Automation., pp 3117-3122(1996)

APPENDIX

The two ICRA Literature Reviews are included below.

Deep Learning for Autonomous Vehicles

Group A Team

Abstract— **Deep learning is used to control autonomous vehicles for velocity dependent collisions in unknown environments. By creating a probability of collision, the computer can choose to put safety or the mission first. In order to allow for dynamic obstacle avoidance, Model Predictive Control is used because it is one of the best controllers for autonomous vehicles.**

Keywords— **Include at least 5 keywords or phrases**

I. Introduction

The car will use a Model Predictive Controller for the autonomous movement. Using Lidar sensors, the car will get data for its surroundings. This can give the position, size, and velocity of each obstacle. Deep neural networks will be used to estimate the probability of collision. This will be directly correlated with velocity input. The MPC controller will control the inputs for maximum comfort of the passenger. It will control the acceleration and steering rate inputs.

II. State Space

The car's pose is described by x, y, and theta position. The car's state is also described by velocity of the car and angular speed of the wheel. The vehicle is non-holonomic and it's important to use the kinematic equations that would represent a car for motion planning.

$$\dot{z}_a(t) = f(z_a(t), u_a(t)) = \begin{bmatrix} v_a(t)\cos(\theta_a(t)) \\ v_a(t)\sin(\theta_a(t)) \\ a_a(t) \\ \frac{1}{L}v_a(t)\tan(\psi_a(t)) \\ \omega_a(t) \end{bmatrix}.$$

Figure 1: Kinematic Equations

III. Control

Deep learning is capable of learning complex interactions and dependencies. The main issue in autonomous vehicle is collision avoidance. They used a MPC for the collision avoidance and the velocity as a dependent for the cost function. They created an independent term for task cost and a function that takes into account collision prediction and velocity.

$$H(t_k, z_a, u_a, o(t_k), h_p) = \int_{t_k}^{t_k+h_p} \gamma_c P_{COLL}(z_a(t_k), u_a(\tau; t_k), o(t_k)) H_{COLL}(z_a(t_k)) d\tau$$

Figure 2: H task equation for Control

PCOLL is the value for collision. When the learning value is uncertain the PCOLL will be higher making the H value higher and lowering the velocity. The optimal control is dependent on the initial state za(tk) and the vehicles observation tk. Hp is the horizon.

When modelling obstacles, they will take the shape of ellipses. They are all assumed to have constant velocities. Each obstacle is modelled as a particle with a coordinate position, a radius and a velocity. This helps turn a dynamic obstacle problem into a static obstacle problem.

IV. Probability of Collision

Supervised learning is usually a hard task with not great results. Having a path set out from the start can cause problems with the smallest of inaccuracies. Using a short horizon and MPC control this can allow for a safer output and less inaccuracies.

The probability is assembled by a group of neural networks. Every neural network by itself is weak because it is hard to visit every possible state but using the group makes it easier and faster to predict the outcome.

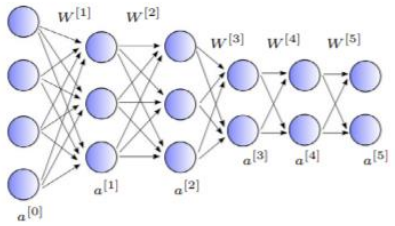

Figure 3: Neural Network

The input layer of the neural network is lidar measurements and the vehicles state and control inputs for each point in the predicted optimal trajectory. The output layer is achieved by applying a non-linear function to a linear combination of

inputs from the last layer. M is the number of layers, W is all the weights of the connecting nodes in jth layers. Each layer uses a RELU linear activation. Each output node has a value of 0 to 1 for the probability of the collision.

$$a^{[j]} = g\left(W^{[j]} \cdot a^{[j-1]T} + b^{[j]}\right), \quad j = 1, \ldots, m$$

Figure 4: Output of every Neural Network Layer

V. Applying to our controller

This proposed way of controlling the car could be beneficial for more unknown environments with obstacles that you don't know that will appear. It seems over complicated for our current scenario. The computation costs are a lot lower with deep neural networks and would make getting a record time a lot easier.

Currently our controller is a PID controller which allows to get around the track at a decent speed. Implementing this controller would make it a lot easier to drift around the corners and specifically reduce time around the corners. Because the track is the same every time around the probability of colliding would be significantly lower than the papers referenced. This would allow the car to go at high speeds around the corners if there was less friction to the ground. The parts of the track with no rails would be a lot harder to speed through and would require a different way of implementing the controller. The car would think there is no way of colliding with anything and therefore try to burst through those sections. This would require the camera to be pretty accurate with its reading and detect how far the edges of the street are from the car to create a wall for a collision.

Overall for this application the deep learning approach wouldn't reduce the lap time by a huge margin because of the simplicity of the obstacles. It would speed up the times around the corners of the track.

References

[1] Sizhao Joe, Qin B. Wayne, Bequette, Lorenz, T. Biegler, Martin Guay, Rolf Findeisen, Jin Wang, Victor Zavala, A Deep Learning Architecture for Predictive Control, 2nd ed., Vancouver, Canada: October, 2018.

[2] Fatemeh Mohseni, Sergii Voronov, Erik Frisk, Deep Learning Model Predictive Control for Autonomous Driving in Unknown Environments, 2018

Human-Robot Interaction research in the Autonomous Vehicles Space

Sai Maddhi
University of Colorado - Boulder
Boulder, CO
sama3612@colorado.edu

Aurangzeb Malik
University of Colorado - Boulder
Boulder, CO
auma9171@colorado.edu

Shreyas Kadekodi
University of Colorado - Boulder
Boulder, CO
shka2478@colorado.edu

Griffin Van Anne
University of Colorado - Boulder
Boulder, CO
grva7970@colorado.edu

Stephen Otto
University of Colorado - Boulder
Boulder, CO
stot7670@colorado.edu

***Abstract*—Autonomous Vehicles, long a sought after goal, have become increasingly prevalent in some form in various industries as well as transportation. As this transition occurs, increasing numbers of humans will come in contact with these autonomous systems. Surveys show a wide variety of responses to the prospect of increasingly autonomous vehicles, varying dramatically based on knowledge, information, and culture. In addition, literature shows that users place a premium on being able to understand the decision making process of the vehicles.**

I. Introduction

Before addressing the research in the field, it is important to note that attitudes towards newly introduced technologies can often change rapidly, [1] with users adapting their behavior in response to technologies in ways they may not have anticipated prior to the technology's introduction. Despite this, it is important to anticipate the public's reactions to new technology for both moral and economic reasons.

A variety of research has been carried out in the field of Autonomous Vehicles and Human-Computer Interaction. However, the research is often of mixed quality and difficult to compare with different sampling methodology and differing questions. For instance, the level of autonomy is not considered in many contexts - certain individuals may consider vehicles that park themselves autonomous, while others may not consider anything less than full autonomy autonomous.

In addition, research also focuses on different aspects of the interaction between humans and vehicles, such as comfort with interactions as well as how to modify robot behavior to ensure the comfort of the individuals in their presence.

II. Survey Results

Surveys have shown that positive attitudes towards Autonomous Vehicles is correlated with exposure to technology within vehicles, such as as driver assistance systems. However, general attitudes towards technology, such as attitudes towards smartphones, are not correlated towards more positive views towards Autonomous Vehicles.

In commercial surveys, respondents tended to be negative towards the idea of autonomous vehicles, particularly when presented with the concept of losing their own decision making ability. However, commercial surveys also tended to be less specific about what "autonomy" meant, meaning users had a broader spectrum of ideas about what that concept meant to them

In surveys that included an in-person discussion and by extension a clearer definition on autonomy or its respective components, it was seen that exposure to existing driver assistance technologies corresponded with better attitudes towards the introduction of autonomous vehicles.

Other surveys showed that users wanted explanations for perceived failures in autonomous vehicles. One such paper gave the example of the Molly problem, an example where a girl named Molly is struck by an unoccupied autonomous vehicle. Other variations of the problem include that the crash was done in an area with few to no witnesses, making direct observations impossible. In the current scenario, investigators would have to painstakingly recreate the accident to see if they could identify the issue, due to the current "black box" nature of the vehicle's decision process. However, both due to regulations and the preferences of individuals who interact with the vehicles, there appears to be more of a push for an explainable decision process. In the above scenario, individuals expressed their discomfort with Autonomous Vehicles if the reasoning for decisions and by extensions the mistakes could not easily be deciphered and corrected. Individuals expressed more comfort with the idea of a potentially fallible human who's intentions and decisions could be deciphered as opposed to an unknowable autonomous system. As a result, much of the published literature in the field of explainable AI stresses the need for Autonomous Vehicles with knowable decisions and transparent processes

Finally, any of the prior surveys noted that in response to changing conditions, users changed their own opinions as well.

However, they noted the moral urgency and economic urgency of addressing issues such as the ones prior in order to allow for a safe and smooth transition to autonomy

.

III. Conclusion

There is much to be explored in the field of Human-Robot interaction with Autonomous vehicles, with different surveys exploring different questions in different ways. However, it is clear that there are a wide multitude of opinions towards Autonomous Robots that will continue to shift as the technology becomes more common.

References

[1] harples, S., Moore, T., Moran, H., Burnett, G., Meng, X., Galea, M., McAuley, D. (2016). Written evidence to the House of Lords Science and Technology Committee AUV0049. Retrieved from http://data.parliament.uk/writtenevidence/committeeevidence.svc/evidencedocument/science-and-technology- committee-lords/autonomous-vehicles/written/41871.html (March 17, 2017)

[2] Bauer, M. (2015). Atoms, bytes and genes: Public resistance and techno-scientific responses. New York: Routledge.

[3] Cohen, T., Jones, P., Cavoli, C. (2017). Social and behavioural questions associated with automated vehicles. Scoping study by UCL Transport Institute. Final Report. Retrieved from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachmentdata/file/585545/social-and- behavioural-questions-associated-with-automated-vehicles-final-report.pdf (January 13, 2017).

[4] T. Schneider, J. Hois, A. Rosenstein, S. Ghellal, D. Theofanou-F ulbier, and A. R. Gerlicher, "Explain yourself! transparency for positive ux in autonomous driving," in Proceedings of the CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–12.