HIGH SCHOOL EDITION
Journal of Student Research

Volume 12 Issue 3 (2023)

Designing Machine Learning Algorithm for Autonomous Driving in the Snow

Divyansh Agarwal

Archbishop Mitty High School

ABSTRACT

Autonomous vehicles have evolved to make decisions with great confidence and accuracy. Systems like the Waymo self-driving clearly can control a vehicle with level 4 autonomy under good conditions where lane markings and other driving guides like road signs are clearly visible [2]. Elon Musk has announced that Tesla autopilot should reach level 4 in 2023 [3]. However, many such systems fail to perform at a similar level in non-perfect weather conditions. The most difficult such weather appears to be a snowy climate as the vehicle significantly underperforms and the autonomous functionality is often not applicable [1]. This paper aims to train a model capable of navigating snowy weather by collecting data pertaining specifically to this weather, including unorthodox visual cues to stand in place of road markings. Such cues include tire tread marks and surrounding snow piles. These serve as a viable substitute for lane markings, allowing the vehicle to operate safely while mimicking human driving tendencies. Such a model can effectively drive in both clear and snowy weather when tested in simulation software. We consider a model effective for the purposes of this study in the snow when it can execute a full turn before failure.

I. INTRODUCTION

Autonomous vehicles (AVs) are growing in their capabilities every year. Tesla's Autopilot is under development and shows promise. Similarly, Waymo has already been running in Phoenix for some time and is working to get its AVs on the road in San Francisco as well [10]. Waymo's self-driving boasts a level 4 autonomy under good conditions where lane markings and other driving guides like road signs are clearly visible [2]. Despite major breakthroughs in autonomous driving, autonomous vehicles today significantly underperform in adverse weather conditions [1]. This paper aims to combat the lack of performance in one specific type of weather: a snowy climate.

As long as there is valid concern about AVs' ability to perform in adverse weather conditions, it remains difficult to deploy autonomous vehicles to many populated areas. This concern stems primarily from AVs' numerous failures to perform in adverse weather, which rightly makes policymakers reluctant to allow AVs to run on the streets [8]. Snowy weather in particular greatly impairs sensors and reduces vehicle control, making it both difficult and careless to deploy AVs in snowy areas without first designing systems capable of combating these limitations [8]. Not only do these limitations prevent the adoption of AVs in snowy areas where they are likely to fail, lack of performance in non-perfect conditions shakes confidence in AVs overall, thus greatly reducing their adoption and usage even in optimal conditions.

Prior work have attempted to alleviate safety concerns regarding adverse weather conditions; all have either chosen not to target snowy weather specifically or decided not to demonstrate their solution through a working model. Prior work have developed effective semi-autonomous solutions to driving in the snow, sensor-fusion for adverse weather conditions, and lane-based autonomous systems; however, the task of a fully autonomous solution for snowy driving remains unfulfilled. This is the main shortcoming we aim to address.

To combat the difficulties of driving in the snow (see Sec. II), we attempt to collect data pertaining to specifically snowy weather, in addition to data with clear skies and roads. Our goal is to train a model that performs well in both clear and snowy weather. The metric we will be using for measuring the success of a model is the amount of time during which the vehicle travels with only safe actions. This paper specifically aims to train a model to understand a city environment both when there is snow falling and clear skies. Furthermore, training data will include some streets that are covered in snow and others that are clear. By using data from both clear and snowy conditions results in a model trained for high performance in both snowy and clear weather.

II. CHALLENGES OF SNOWY DRIVING

Driving in the snow has always been challenging for vehicles and drivers. Drivers suffer from reduced visibility as collected snow often blocks key visual cues such as lane markings, road signs, crosswalk markings, intersection turn guides, and the end of the road. Falling snow can obstruct vision and make it harder for drivers to see their surroundings and make the correct judgments. The road is also usually more slippery when it's covered by snow, making it harder to control the vehicle even if one's sensory perception remains unimpaired. There always exists the danger of sliding or skidding on the road as well as losing control entirely. Furthermore, on roads that lack regular maintenance, there is a possibility of vehicles getting stuck in the snow.

There exist more problems for autonomously driving in snowy weather. Key sensors like LiDAR are at least partially obstructed if not rendered completely useless in major cases and are less reliable even in mild snow [6]. Falling snow particles can interfere with the light that is emitted and reflected back into the LiDAR sensor. As such, the autonomous vehicle can no longer completely rely on its primary sensor, hindering driving capabilities in the snow [1]. In general, most sensors that function by emitting and receiving something (light, sound, etc.) will suffer from reduced performance in the snow due to obstruction from the falling snow.

Further difficulties are illustrated through the first, naive, model that was used. This model used the baseline architecture (see: Sec. IV-A) but with a larger input shape for raw camera data instead of camera data processed by a CNN (Convolutional Neural Network). The model correctly turned away from the wall when it closed in on it but was incapable of determining the correct direction of travel. In fact, driving straight at an angle rather than parallel to the road was the behavior most of the time. Essentially, the model failed to note differences in the directions of travel because of the landscape seeming to appear rather uniform in the mountainous snow environment that this model was tested in. This highlights the uniformity of snow as another problem since machine learning models use distinct features to draw patterns from data.

III. PRIOR WORK

Huval et al. [7] used a CNN and sliding window for lane and obstacle detection. They highlight each lane marker in the road to plan a trajectory for the AV. Such a method is effective in clear weather; however, under snowy weather conditions, the lane markings are often not visible, making this method inapplicable. Should the algorithm not be able to see the lane markings, the AV will lose its autonomous function on the road. The CNN/sliding window technique would need to be repurposed to other data relevant to snowy driving. If this is done, it can allow AV to do similar classification of its surroundings.

Fridman et al. [5] attempted to use an arguing machines concept to decide when human intervention was necessary. The system works by comparing the Tesla autopilot with an end-to-end neural network. When the two machines "disagree," the system cannot be confident in either network's decision and must have the human take over. While this has the potential to alleviate some safety concerns of AVs in adverse weather conditions by notifying the human supervisor when it is necessary to take control, it does nothing to actually drive autonomously in the non-optimal snowy conditions. This is an effective semi-autonomous system that increases the possibility of deploying AVs in the snow; our aim is to create a fully autonomous system. If there is snow covering the road, the arguing machines will notify the driver that they must take control of the vehicle, but the vehicle itself cannot operate autonomously. Therefore, more work is needed to develop a full-autonomy solution.

Specifically in adverse weather conditions, Yeong et al. [11] discuss the degradation in sensor performance and offer an analysis of sensor fusion techniques to combat said decrease in performance. For example, they discuss the possibility of overlaying radar or LiDAR over camera feed. A fusion of these two sensors can help combat the obstruction of radar/LiDAR and the lack of clarity in the camera feed. However, this study does not go as far as creating a system capable of autonomous driving in adverse weather conditions.

IV. SYSTEM DESIGN

Given the challenges of driving in the snow, training our model required the use of unorthodox data. In the absence of lane markings and dividers, the task of identifying where the AV should be positioned required training on data that is not normally required for regular city driving. One of the unique challenges of snow is that it collects on the road, and in sufficiently cold climates, it does not melt. Although this is a large problem for driving, it also presents an opportunity as, to maintain drivable conditions, the snow is often pushed to the side of the road. Identifying the position of the collected snow allows us to train the model on this data, as there is a direct relationship between the sides of the road and where humans drive. In smaller areas, this technique can be highly effective in determining the correct vehicle placement. However, in well-maintained areas, especially big cities, often the road is not fully covered in snow and the side of the road does not have large snowbanks.

There are still elements of the snow we can use to our advantage. The sidewalk will likely be highly white with collected snow, pointing us to the edge of the road. Furthermore, with less snow present, the task becomes more akin to how currently existing algorithms discern the side of the road. The snow masks features that existing algorithms may use, but there will likely still be buildings beyond the sidewalk that can also serve as a method of judging the AV's position.

One network is dedicated to simply finding these collected snowbanks or the edge of the road. Eventually, it becomes more effective for this model to simply determine a state vector with 4 values that represent the lane placement and distance from intersection. The data is then fed into more machine learning algorithms to predict throttle, steering, and brake values. These three values are the required control inputs for the vehicle inside simulation software.

A. Baseline Model Architecture

The baseline model was simply a Sequential Neural Network. The network takes in state input to select the optimal control inputs for the vehicle. In the initial test, this state input was image data in the form of an array, distance sensor data, LiDAR data, and current speed so the input layer had 113594 neurons. This is significantly reduced in the second iteration where another model passes in processed image data so the input layer was 3006 neurons. The remaining part of the network was common to both iterations: 2 dense layers of 128 neurons and an output dense layer with 3 neurons (one for each control input to the car).

B. CNN Architecture

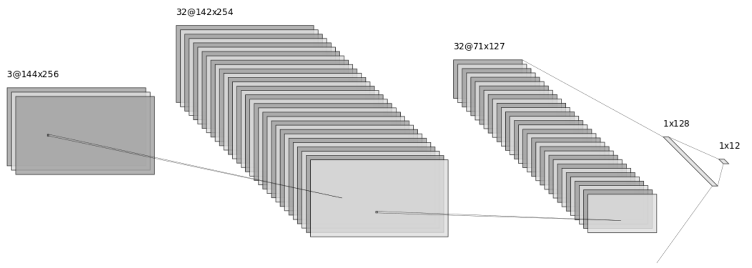

Fig. 1. Image processing CNN, road edge detection (version 1)

The CNN architecture varied as development continued; however, initially the most promising model seemed to start with a 2D convolutional layer with an input shape of 144 x 256 x 3 (the dimensions of the image) and 32 3x3 kernels. Following this was a 2D max-pooling layer of pool size 2x2. The following layers in order are a flattening layer and a dense layer of 128 neurons. The final output layer was initially of size 12 to predict the coordinates of the edge of the road.

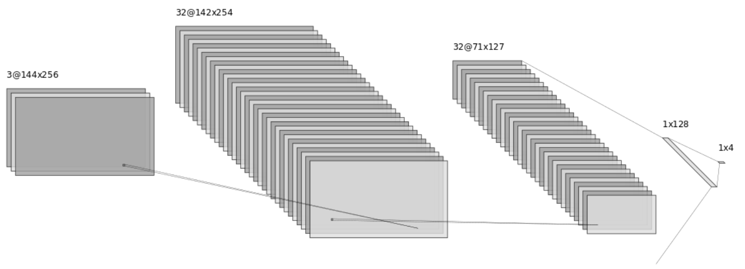

Fig. 2. Image processing CNN, robot state prediction (version 2)

However, it was more effective to have an output dense layer of 4 neurons representing information about the vehicle's placement on the road and the distance from intersection. The first value represents whether the vehicle is centered in the road, and the other three are a one-hot encoding of the values far-from-intersection, arrived-at-intersection, and in-intersection. Even more effective was the usage of transfer learning in the CNN. The most effective model started with VGG16 with an input shape of 144 x 256 x 3 (image dimensions). This was followed by a 1024 neuron dense layer and ended in the same 4 neuron configuration as the other model.

C. Reinforcement Learning Architecture

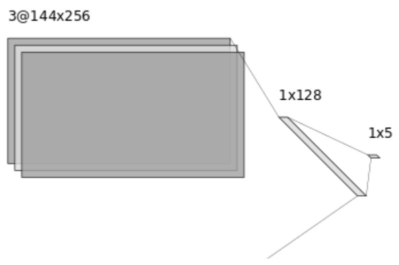

Fig. 3. Reinforcement learning model, image data to action prediction (version 1)

The first iteration of the reinforcement learning model was an actor-critic network in which both the actor and critic took in a 144 x 256 x 3 image as input. This is followed by a 128-neuron dense layer common to both networks. Finally, the actor network ends with a 2-neuron dense layer that corresponds with the choice to steer sent to the vehicle and the critic network ends with a 1-neuron dense layer that corresponds with the perceived reward.

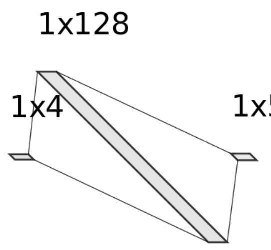

Fig. 4. Reinforcement learning model, state prediction to action prediction (version 2)

The final model used was also an actor-critic reinforcement learning model. This time, both the actor and critic networks start with a 4-neuron input layer that takes in state data from CNN. The remainder of the networks is identical to the first model.

D. Training Process

The first method's training process starts with recording data in the AirSim. AirSim gathers state data about the car, such as speed, as well as recording image and LiDAR data right from Unreal Engine where the environment is built and simulated. The first iteration of the CNN predicted the location of the ends of the road. Image data can be labelled with the boundary points of the sidewalk which is then used to train the CNN (see Sec. IV-B) to recognize the edges of the road. This data needs to be passed into the base model (see Sec. IV-A), so once the CNN is trained, it is used to prepare the rest of the data before training this second model. This data is used to train three models; one is for turning left, the second for going straight, and the third for turning right. These three models are trained on the 6 input coordinates for the borders (spanning over 12 neurons), 1000 LiDAR points (spanning over 3000 neurons), and the current speed of the car. There are three separate data sets of this structure, one for each model.

However, there was a more effective method than predicting the borders of the road. Instead, it turned out to be easier to simply predict the ”state” the vehicle was in at the time. This state is a vector with 4 values representing lane placement and distance from intersection. We labelled the data with the first value in the vector being either 1 or 0 for centered in the lane and not centered in the lane respectively. The next three values in the vector represent the possible states: far from intersection, arrived at intersection, and in intersection. When collecting data to be used in this way, it was important to ensure that there was no bias towards any particular state. This entailed having a very similar number of data points for the three different distances from intersection even though in real driving the far-from-intersection state is easily the most common. Eventually, the baseline model and actor-critic model were designed to take this new state in as input rather than the border points (see Sec. IV-A, Sec. IV-C).

Training the actor-critic model was different. The Python training script links to the Unreal Engine environment and uses the AirSim Python API to control the vehicle in real time. The model outputs probabilities for each action in a set of 5 actions. These actions are sent to the AirSim as control inputs for the vehicle. The model uses the performance from each episode (attempt to complete a task) to decide how to adjust the weights. As such, all processing of the state data and predictions on what actions to take were all made in real time, similar to how the model would eventually be applied after its completion.

All training was completed by collecting data as described and uploading it to Google Drive. From there it can be accessed by Google Colab, where it is processed and saved back to Google Drive. Since this data can now be accessed by Colab, it can be loaded into a notebook and used to train the models (see Sec. IV-A, Sec. IV-B, Sec. IV-C).

V. EVALUATION

A. Methodology

Our method for testing and training the algorithm relies on the use of simulation software. The chosen simulation software was Microsoft's AirSim. This software allows for easy weather changes and has a good physics engine that can be used right out of the box. Said weather changes allow for diversity in data as some data can be collected with it actively snowing, and other data can be collected with clear skies. The level of snowfall is highly customizable, allowing the user to select a percentage. AirSim's environment is built in Unreal Engine, so it allows users to configure their own environments; as such we use Blender and Unreal Engine to set up our custom snowy environments.

AirSim contains all necessary sensors, and our model was trained on camera feed, LiDAR, distance sensor, and current speed of the vehicle. Said camera data was annotated with the snowbanks or edge of road mentioned in Sec. IV. Afterwards, the camera data was annotated with the state data used to train the new models (see Sec. IV-D). LiDAR offered a full 360 view capped to 1000 data points. All these forms of data were collected via AirSim's Python API during human driving in the Unreal Engine environments. All models were built with Google's Keras library and trained in Google Colab except for the actor-critic network since it needed to be trained locally with the simulator.

We use the built in TPU that comes with the Google Colab Pro subscription to train our CNN and baseline models. The actor-critic network was trained locally on a laptop with an Intel i7 10th gen CPU and Intel Iris Plus graphics.

B. Model Performance

Table 1. The performance of each model as a length of time without departing from the road

Model	Travel Time Without Error
Baseline	5 seconds
Baseline+CNN	11 seconds
Actor-Critic+CNN	27 seconds

1. Baseline Model: The baseline model performed terribly despite having the easiest environment. It was trained to simply stay between the snow bumps and travel parallel to the road. It did not end up learning this effectively; it drove at an angle, rarely parallel to the street. It did try to self-correct just before hitting the wall, but it never effectively avoided the collision. This model's poor performance was likely due to using only dense layers, which are not very effective at capturing elements of image data.

2. Baseline+CNN: After adding CNN processing above the baseline model, it seemed to perform somewhat better. However, it still often failed to perform well in real situations, seeming almost ”confused.” This is likely explained by the fact that the models were all trained through supervised learning. This implies that there is only one ”correct” way to handle a given set of inputs and offering a full range of inputs becomes difficult without offering more guidance on specific elements to be filtered out. Since, in real driving, there is a large amount of variation in outside conditions (input to the model), it seems ineffective to train it for one driving style on certain inputs rather than letting it learn more “organically.”

3. Actor-Critic+CNN: Reinforcement learning of an actor-critic network is exactly the form of training that allows for a more "organic" training. This model performed the best of the group. It was clear that the model was drawing patterns between the state and the desired action. The model was provided with "good" regions and "bad" regions to travel in. The model was trained to prioritize being in a "good" region; these regions draw out the path of an action at an intersection. For example, when training the model to turn left, the "good" regions together draw out a left turn. By learning to stay within these bounds to maximize reward, the model effectively learned the sequence to make a left turn. Given imperfections in the CNN state prediction, it stands to reason with better state input, the actor-critic network would work even better.

C. Interpretation of Results

In addition to the actual time data, it is quite obvious that the model reacts to certain stimuli when driving. For example, the reinforcement learning model clearly shifts left in the lane before increasing the rate of turn to make the left turn inside the intersection. This reflects an understanding of the environment despite the snowy obstructions, signaling the correct time to execute the turn. During this leftward shift, the path is very straight. Only after the vehicle reaches the intersection does it start to turn rapidly. The turn does seem choppy with the vehicle switching between having a very large angular velocity at times and appearing to be driving straight at other times, but generally executes successfully, thus meeting our initial goal for a successful model. However, its overall quality of control is poor enough that the model fails soon after by crashing.

Simply following the road while understanding its surroundings seemed to be a tough task for both the Baseline+CNN model and the Actor-Critic+CNN model at times; however, the Baseline only model had even more trouble, failing at even this basic task very quickly. In fact, regardless of the situation, the Baseline-only model failed very quickly compared to the other models—adding a CNN doubled the performance and switching to reinforcement learning more than quintupled it.

In executing each task, the models also had varying consistency to go along with a change in performance. The Baseline-only model was very sensitive to slight changes in input, having wildly different behavior. Removing direct dependency on inputs by adding a CNN that provides a state that the Baseline model can use helped to limit its erratic nature. However, since both these models still use supervised learning, they are still sensitive to variations in training data that can manifest in the final control of the vehicle. This could be remedied by simply including more diverse data points, but a representative sample of camera feeds in particular would be nearly impossible. Hence, the use of reinforcement learning allowed for far more consistency than either of the other models: the larger flexibility in acceptable output allows the model to operate in a similar manner despite potential differences in input from episode to episode.

VI. DRIVING WITH HUMAN INTERVENTION

In the period of time before AVs are fully able to navigate all forms of snowy weather, it would be beneficial to be able to deploy autonomous systems in areas with non-ideal weather conditions to track their successes and failures. Furthermore, autonomous systems have been proven to be safer than humans overall [9]. Allowing AVs to continue to run in adverse conditions like snowy weather would still be useful in keeping people safe. However, their limitations in these adverse conditions prevent them from driving alone. As such, a human operator would still need to be present. This would offer developers valuable information about the performance of their systems in adverse weather conditions while also helping keep people safe.

However, there are also dangers that come with implementing such technology. Namely, people being lulled into complacency and not paying attention to the driving task. This has been explored for level three autonomy [4], and it stands to reason that something similar can occur for autonomous systems in adverse weather conditions. As such, implementing this technology would likely need to be complemented by monitoring methods to detect if the driver is paying attention. If the monitoring system detects that the driver is not paying attention, it should deactivate the autonomous functionality.

It stands to reason that with such defensive measures, deploying such systems early is the best course of action—developers get valuable information while preserving safe driving for the human sitting inside.

VII. DISCUSSION

A key takeaway from the training process (see Sec. IV-D) is that passing in image data as an array is highly ineffective. Convolutional layers perform far better for analyzing images, although they require a lot of data as evidenced by the generally poor performance of the CNNs. In fact, given that CNNs are the bottleneck in effectiveness and provide vital state data, having enough image training data for the CNNs is extremely important to future success. Another problem was that using supervised learning restricted the network to only one way of doing things which prevented the model from performing well. This was improved upon by the reinforcement learning model which resulted in far more success. Considering that the reinforcement learning model allows the network to learn more organically for a task that has many solutions, it is not surprising that it was more effective.

This project could easily have been improved through better training of models, especially the CNNs. All models were trained in Google Colab and locally on a laptop, so more computationally intensive fine tuning of networks was not feasible. With stronger hardware like deep-learning servers, deeper fine tuning of the models would be possible—this is one effective way to better train the models, which was the main flaw in the system. Another potential improvement to the training process would be to record more data. Furthermore, in the initial data collection practices, some outputs were far more common than others, incentivizing the network to get better at predicting based on the relative size of subsets rather than by drawing out patterns in the data itself. Because of this, future data for future CNN training was collected with this in mind. Another thing to experiment with would be the network architectures and transfer learning models used to train a model more capable of recognizing features in training data. For example, the model with VGG16 architecture (see Sec. IV-B) seemed to be most effective, but there could easily be other architectures that perform better for this problem.

Any future work that builds upon the reinforcement learning strategy should invest in good state prediction. In the case of image data this will likely be a CNN. If the future work uses reinforcement learning as well, it would be key to have good state input (a good CNN). Another potential avenue to consider would be using a verification model. This model would be trained not to predict the correct action based on the state but verify that a certain action given the current state is a “good idea.” Training such a verification model would likely entail taking recordings of manual driving and labelling frames as either helpful actions or hurtful actions. Furthermore, one intrinsic weakness of the reinforcement learning model is in its Markov chain decision making; however, driving is not an activity that always only relies on the current state, but it can also depend on previous actions which are completely ignored. Future work could attempt to target this weakness by passing in multiple previous frames instead of just the current state as input.

VIII. CONCLUSION

We learn from this work that collecting data specifically pertaining to a snowy environment is an effective way to boost performance in this weather, but it is highly dependent on what model architecture is used and what training practices are observed. This work also reaffirms the idea that reinforcement learning is effective in actions like driving or playing video games in which there are many possible routes to a successful result. In any case, understanding the vehicle’s surroundings is a priority, showing the importance of good state analysis (CNNs for images, etc.).

REFERENCES

[1] A. C. M. (AccuWeather), *How can self-driving cars 'see' in the rain, snow and fog?* Jan. 2021. [Online]. Available: https://www.abc10.com/article/weather/accuweather/self-driving-cars-radar-inclement-weather-rain-fog-snow/507-0438604e-ef32-4c0a-9634-99a6ec71fa12

[2] E. Ackerman, "What full autonomy means for the waymo driver," *IEEE Spectrum*, vol. 4, 2021.

[3] G. Davis, *Elon musk: Tesla full autonomy promised this 2023, again*, Apr. 2023. [Online]. Available: https://www.techtimes.com/articles/290617/20230420/elon-musk-tesla-full-autonomy-promised-2023-again-referring-level.htm

[4] L. Eliot, *Distracted driving will grow exponentially on the path to self-driving cars*, May 2019. [Online]. Available: https://www.forbes.com/sites/lanceeliot/2019/05/16/distracted-driving-to-growexponentially-on-the-path-to-self-driving-cars/?sh=660472b41c1c.

[5] L. Fridman, L. Ding, B. Jenik, and B. Reimer, *Arguing machines: Human supervision of black box ai systems that make life-critical decisions*, 2017. DOI: 10.48550/ARXIV.1710.04459. [Online]. Available: https://arxiv.org/abs/1710.04459.

[6] *How will self-driving cars fare in bad weather?: Endurance*, Jan. 2021. [Online]. Available: https: //www.endurancewarranty.com/learning-center/tech/self-driving-cars-bad-weather/.

[7] B. Huval, T. Wang, S. Tandon, *et al.*, *An empirical evaluation of deep learning on highway driving*, 2015. DOI: 10.48550/ARXIV.1504.01716. [Online]. Available: https://arxiv.org/abs/1504.01716.

[8] W. Knight, *Snow and ice pose a vexing obstacle for self driving cars*, Feb. 2020. [Online]. Available: https://www.wired.com/story/snow-ice-pose-vexing-obstacle-self-driving-cars/.

[9] G. Nash, *Are self-driving cars safer than human drivers?* Apr. 2022. [Online]. Available: https: //www.way.com/blog/are-self-driving-cars-safer-than-humans/#:~:text=In%20certain%20aspects% 2C%20self%2Ddriving,under%20the%20influence%20of%20alcohol..

[10] *Waymo one*. [Online]. Available: https://waymo.com/waymo-one/.

[11] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, 2021, ISSN: 1424-8220. DOI: 10.3390/ s21062140. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2140.