Perception-based Autonomous Driving in a Simulated Crowded City

Baran Ekin Özdemir, *Techical University of Munich*

***Abstract*—The development of self-driving cars is a game-changing step in transportation technology that can revolutionize how we travel in the near future, hence is one of the hottest topics in today's research. This project is offered by the chair of Robotics, Artificial Intelligence and Real-Time Systems of TUM CIT and it seeks to address the task of developing a robust autonomous driving system within a rather complex 3D city simulation. The core challenge revolves around using the data from the ego car's optical cameras to ensure it can make safe and accurate driving decisions, maneuvering through the simulated urban streets while avoiding collision with other objects.**

I. Introduction

A. Problem Statement

In this project, we confront the challenge of achieving autonomous driving within a 3D simulated city environment. This simulated urban landscape encompasses a network of streets, buildings, trees, and various vehicle agents controlled by the simulation such as cars and busses. Our primary objective is enabling an ego car to navigate these city streets safely and independently, relying exclusively on its perceptual capabilities. Specifically, the ego car relies on data from three optical cameras positioned to capture its forward surroundings (left, center, right). Notably, this simulated environment excludes the presence of pedestrians or bicyclists and does not involve a predefined destination or path planning. Navigation, GPS and SLAM is out of scope of this project.

B. Motivation

The development of self-driving cars is a significant effort that could change how we travel in a really good way. In today's world, our roads face problems like traffic jams and accidents that are due to human error more often than not.

One of the most important things self-driving cars can do is make driving a lot safer. They can be designed to adhere all the traffic rules perfectly and avoid accidents caused by human mistakes, for example like not paying attention or breaking traffic rules. This means fewer crashes, fewer injuries, and fewer lives lost on the road. It's like having a super-smart driver who never gets tired, distracted, or impatient.

Even though being most important aspect, autonomous driving is not just about making the driving safer, it can also make it more convenient for people who can't drive because they are too old, too young, have a disability, or other reasons. Self-driving cars can give them the ability to go where they want when they want, without needing someone else to drive them. This is a big deal for their independence and quality of life.

Moreover, self-driving cars can be better for the environment by allowing vehicles to be co-owned, reducing number of vehicles on the road. Also, they can help reduce traffic jams by driving more efficiently, which is good for the environment and our overall well-being.

II. Method

A. Main Architecture and Logic

As visualized in Figure 1, the method consists of two different deep neural network models, first being the steering model and second being the vehicle detection model.

Steering model takes in the camera input and estimates a raw steering value. This value is later adapted via steering adaptation to remove steering bias due to using mouse as a controller in simulator during data recording sessions and to adapt vehicle velocity from 30 km/h during training to 20 km/h during testing in order to safely drive and stop on time when required.

Vehicle detection model takes the same camera input as steering model does and outputs vehicle detections. These detections than fed to the collision checking function to decide whether the ego car should stop, slow or continue.

Finally, vehicle control algorithm takes steering and collision check signal into account to produce control outputs (steering, throttle and break) to ensure vehicle is kept at the velocity of 20 km/h and can safely drive through the streets avoiding collision.

B. Dataset Acquisition

The simulator has a mode for data capturing where human player can drive the ego car either by keyboard, mouse or controller while frames from each camera are being saved as a single image, along with current car velocity and control inputs such as steering, throttle and brake.

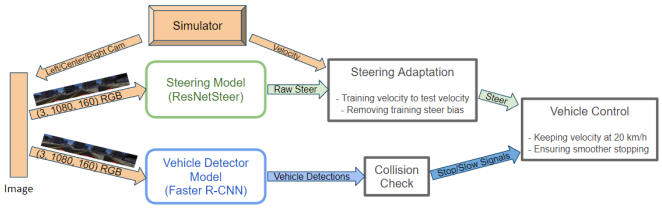

Fig. 1. Visualization of the architecture of the method. Two networks work together to drive the ego car via control outputs to the simulator.

The primary objective was to construct a robust dataset to facilitate the training of our deep learning steering model, ultimately geared toward the realization of self-driving capabilities within provided simulated environment. In order to achieve this goal, a final dataset comprising a noteworthy quantity of approximately 86,000 images was acquired in hours of manual driven recording sessions.

Throughout the duration of these data capturing sessions, mouse is used rather than keyboard for control input. This approach was to ensure the generation of smooth and continuous steering values, mimicking the nuanced steering behaviors akin to human driving with an actual steering wheel in real life. Having discrete steering values created by keyboard controls generates a dataset where consecutive images are very similar, however steering values being far apart, especially when pressing and releasing the keys. Such dataset is not suitable for training.

During the data capturing sessions, a deterministic approach was followed. For example, In contrast to having randomized decisions, our vehicle adhered to a straightforward rule, namely, maintaining a straight trajectory at intersections, unless the road layout itself necessitated a turn. This strategic choice was made to enhance models confidence during intersections. If dataset consists of different turning decisions at intersections, model learns to approximate the mean of such diverse decisions, going towards non-drive-able surfaces.

It's important to underline that, as part of our data capturing process, our simulated vehicle did not adhere to conventional traffic regulations. This deviation was a conscious decision to expedite the data collection process and simulate an efficient and focused approach to steering decision aspect only, eliminating any unnecessary wait times behind other vehicles in the virtual environment. Also, vehicle velocity is kept at the maximum speed limit set by the simulator which is 30 km/h, to cover more area as fast as possible by constantly applying full throttle without any downside since the model only trains on steering values.

C. Steering Model

In our project, realization of autonomous driving in our simulated environment relies significantly on the collaborative efforts of two core deep learning models. The primary one among these models is the Steering Model, which serves

Fig. 2. An example image from the captured dataset. Three cameras jointly create 1080x160 RGB image to be used as an input to the method.

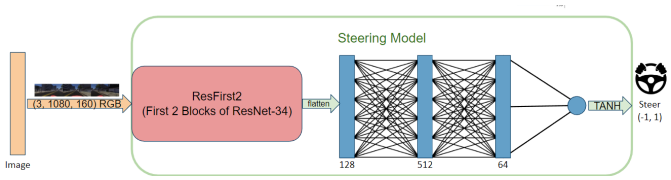

Fig. 3. Visualization of the architecture of the Steering Model.

as the main component of the entire system. This model's task is processing a singular 1080x160 RGB image which is simultaneously obtained from putting all three available cameras side-by-side, and subsequently producing a steering value in the range of (-1, 1) as its output. It is tasked with the critical responsibility of interpreting multi-camera visual data to make precise steering decisions within the simulated environment to keep the ego car on it's lane.

1) Architecture: The Steering Model adopts a ResNet [1] architecture as its foundational backbone. Specifically, it leverages ResFirst2, which constitutes the initial two blocks, encompassing 15 layers, of the ResNet-34 architecture. -ResFirst2 does not involve pre-training as the simulated environment is a novel challenge.

The Steering Model extends beyond the ResFirst2 by incorporating an additional three fully connected layers after the ResFirst2 layers. In order to achieve an output range within the bounds of (-1, 1), TanH activation function, short for "Hyperbolic Tangent" is used. TanH is a bounded activation function that maps its input values to fall within the range of (-1, 1), which allows us to transform the model's predictions into a format compatible with steering control of the simulator.

2) Res-Net Backbone: ResNet [1] , short for "Residual Network," represents a deep Convolutional Neural Network (CNN) design renowned for its capacity to overcome challenges related to training very deep neural networks. It was introduced to address a fundamental challenge faced by deep neural networks – the vanishing gradient problem. This issue arises when gradients, which guide the model's learning process during training, become exceedingly small as they propagate backward through the many layers of a deep network. As a result, the network struggles to learn and adapt effectively, impeding its ability to capture intricate features in data.

ResNet tackles this problem by incorporating skip connections or shortcuts, also known as residual connections, which enable the direct flow of information from one layer to another, bypassing intermediate layers. This architectural innovation allows the network to learn residual functions,

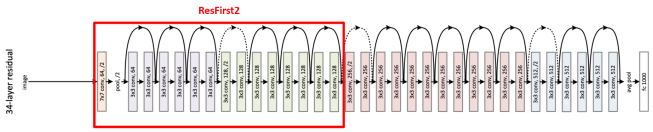

Fig. 4. Visualization of the ResFirst2, which is the first two blocks of ResNet-34. ResFirst2 is the backbone for the Steering Model.

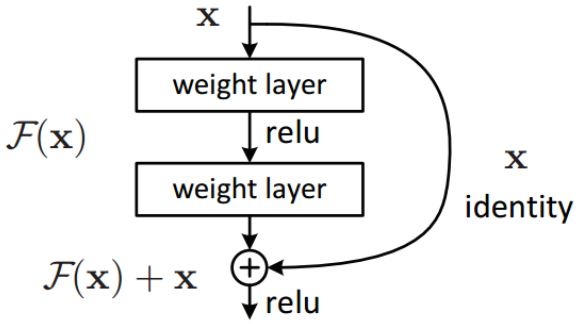

Fig. 5. Visualization of the ResNet block with residual connection that allows gradients to flow to prevent vanishing gradient problem.

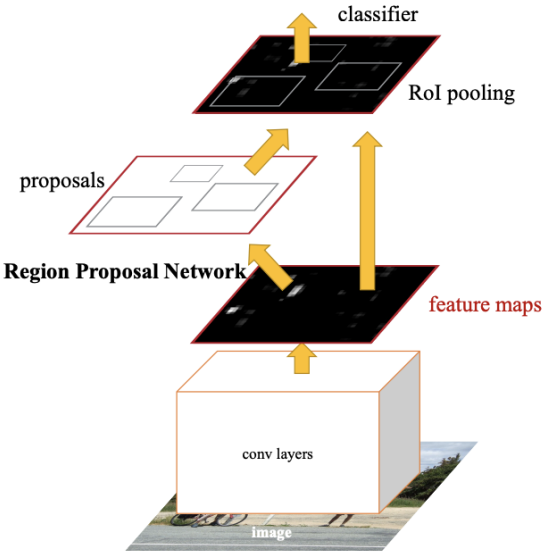

Fig. 6. Visualization of the architecture of Faster R-CNN. Conv layers in this case is the MobileNetV3 network.

which are the deviations from the identity mapping, making it easier to optimize very deep networks.

Consequently, ResNet models, like our ResFirst2, can effectively harness their depth to extract complex hierarchical features from images while avoiding the vanishing gradient dilemma. In this case, ResFirst2 is used to keep network size small and achieve better generalization of the available data.

D. Vehicle Detection Model

The second component, the Vehicle Detection Model, plays a vital role in ensuring safe autonomous driving within an environment populated by other vehicles. While the Steering Model is responsible for controlling the vehicle's direction, it is imperative to also detect and respond to the presence of other vehicles to prevent collisions. To address this crucial aspect and to ensure safe driving, a pre-trained Faster R-CNN (Region-based Convolutional Neural Network) [2] is employed as the Vehicle Detection Model.

The Faster R-CNN model is specifically chosen because it was readily available in the torchvision library as a pre-trained model. It has been pre-trained on the COCO [3] (Common Objects in Context) dataset. This dataset encompasses a broad range of object classes, including car, truck, and bus that are related to this project's objectives. No further training of Vehicle Detection Model was done for this project.

1) Object Detection Task: Object detection stands as a very important component within the objective of autonomous driving, addressing the fundamental task of identifying and localizing objects of interest within the vehicle's environment. It is the process of locating and categorizing various objects, such as vehicles, pedestrians, road signs, and traffic lights, within the visual field of the vehicle's sensors, typically cameras. This technology plays an important role in creating safe and effective self-driving systems, enabling the vehicle to make informed decisions based on its perception of the surrounding world.

For this project, object detection, more specifically vehicle detection, is used in order to avoid collision with other vehicles in the simulator, ensuring that the ego car can safely drive around.

2) Faster R-CNN with MobileNetV3 Backbone: R-CNN's (Regional Convolutional Neural Networks) are a family of multi-stage detectors. One of the notable advancements offered by Faster R-CNN over the original R-CNN is the integration of a Region Proposal Network (RPN) within the model itself. This innovation eliminates the need for external proposal generation, resulting in significantly improved speed and real-time capabilities. Within this framework, the model first generates region proposals, which are potential object locations in an image. These proposals are identified and refined through a RPN, which operates in tandem with the feature maps produced by the convolutional layers. The RPN uses different scales and aspect ratios, offering a diverse set of candidate regions for subsequent object detection. Following the proposal generation phase, the selected regions are then subjected to a final object classification and bounding box regression, achieved through dedicated convolutional layers. This multi-stage approach enables Faster R-CNN to excel in both accuracy and speed, positioning it as a highly effective choice for object detection tasks, including the need to detect and localize other vehicles in our autonomous driving simulation.

Moreover, Faster R-CNN offers flexibility in its choice of

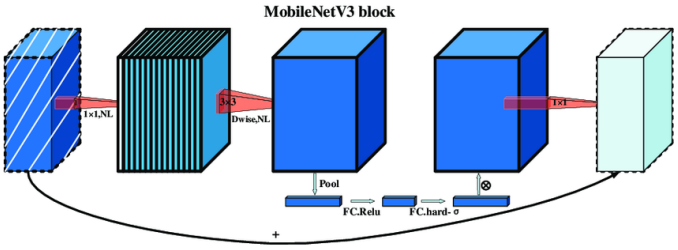

Fig. 7. Visualization of the architecture of MobileNetV3 block

convolutional backbones.In this context, MobileNetV3 [4] , renowned for its exceptional efficiency and designed for mobile and embedded devices, is employed as the backbone of the Faster R-CNN model.

MobileNetV3 achieves this efficiency through a combination of innovative techniques. The main standout feature is the use of depth-wise separable convolutions, which split convolution into separate depth-wise and point-wise (1x1) convolutions. This separation significantly reduces computational complexity while preserving model performance. Additionally, MobileNetV3 incorporates channel-wise and spatial-wise feature fusion mechanisms, allowing for the efficient integration of information from different network layers. These features collectively ensure that MobileNetV3 maintains a good computational cost and detection accuracy, making it an ideal choice as the backbone for our Faster R-CNN-based Vehicle Detection Model.

3) Vehicle Detection in Run-time: During the testing time of method, Vehicle Detection Model makes object predictions for all camera images as a single input image. Predictions with confidence score below 0.7 are discarded along with the predictions with classes other than car, bus and truck.

Depending on the center point and width of each object box, collision checking algorithm decides whether to slow, stop or continue. This slow, stop and continue signals are then propagated to main driving logic to produce throttle output, which is also dependent on the current velocity of the ego car.

III. Experiment and Results

A. Training

To train the steering model, final dataset comprising a total of 86,000 images, divided into a training set of 80% and a validation set of 20%. This division allows us to train our model on a substantial portion of the data while reserving a smaller segment for ongoing evaluation, ensuring that our model generalizes well to unseen scenes.

Since the city map of the simulator is small and there are few roads, novelty in this dataset usually comes from positions of other vehicles in the traffic. Regarding this, to achieve a better generalization a technique known as "Random Erasing" is employed as part of our data augmentation strategy. This method involves randomly deleting portions of the input images and replacing them with either 0, 255, the image mean, or random noise. The primary objective of this augmentation is to simulate scenarios where other vehicles occlude parts of the camera's field of view. By introducing this variability into the training data, we equip the steering model with the capacity to handle situations involving partial occlusion during intense traffic -especially at intersections-, enhancing its overall robustness and adaptability to a wide array of different scenarios.

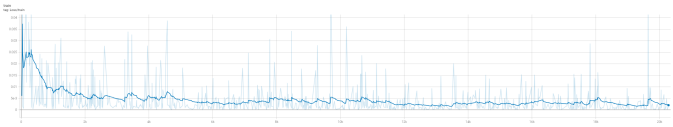

Fig. 8. Training loss over two epochs. Line in the middle is the smoothed value as it is more representative of the training since there is high noise in training loss due to small batch size.

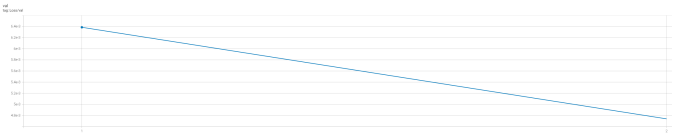

Fig. 9. Test loss over two epochs. Training is terminated after two epochs to overcome over fitting to the small dataset.

The training process is executed over two epochs, with each epoch representing a complete pass through the entire dataset. During each epoch, the data is processed in batches of eight images. Adam optimizer, a popular choice in deep learning, is used with a learning rate set to 1e-4. The optimizer's role is to adjust the model's parameters during training to minimize the steering prediction error.

As a loss function, Mean Squared Error (MSE) in between predicted and actual steering value is used.

$$\sum_{i=1}^{D}(x_i - y_i)^2 \tag{1}$$

B. Simulator Results

At the test time in the simulator run, method is able to navigate the car safely around the city without any collision with other vehicles. Figures 10, 11, 12, 13 show example occurrences during the simulator run.

IV. Conclusion

A. Limitations

Our project, while promising and innovative, does come with certain limitations that warrant careful consideration.

Firstly, it's important to note that this approach does not incorporate Simultaneous Localization and Mapping (SLAM) or GPS mapping as it is out of scope of the simulator

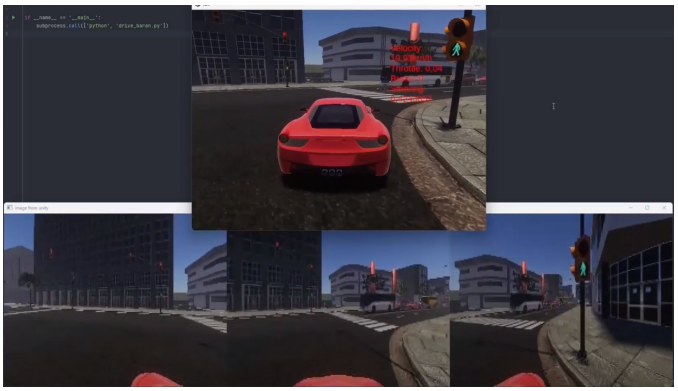

Fig. 10. Screenshot from the simulator at test time, during a successful right turn. Above image shows the simulator and below is the camera left, center and right camera input to the method in real-time.

Fig. 11. Screenshot from the simulator at test time, during a successful left turn. Above image shows the simulator and below is the camera left, center and right camera input to the method in real-time

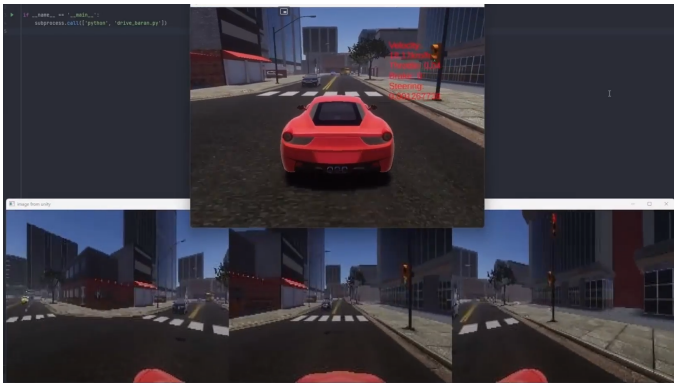

Fig. 12. Screenshot from the simulator at test time, during an intersection, going straight. Above image shows the simulator and below is the camera left, center and right camera input to the method in real-time.

Fig. 13. Screenshot from the simulator at test time, demonstrating the ability of the ego car to safely stop behind another vehicle, in this case a long bus. Above image shows the simulator and below is the camera left, center and right camera input to the method in real-time

capabilities. As a result, our simulated vehicle adheres to a relatively simple path-following strategy, continually going straight until it is required to make a turn. This approach, while functional to a certain extent, results in the vehicle going through the same areas repeatedly if manual intervention is not applied.

Secondly, the method faces challenges in the domain of traffic light detection and classification. The output resolution of the simulator, which this project relies on for environmental perception, is very limited. This limitation causes traffic lights to often remain indistinguishable at significant distances within the simulated environment. As the ego car is not able to accurately detect and respond to traffic lights in a timely manner, it is potentially disregarding traffic rules and putting itself at risk of collisions at intersections.

Finally, within our simulated environment, a notable limitation is in the behavior of other vehicles at intersections. These simulated vehicles do not consistently follow turning rules, often attempting to execute right turns from the left lane or left turns from the right lane. This erratic behavior often leads to direct collisions with the ego vehicle from the side, which can not be receipted by the camera angles, changing ego car's intended course and undermining the realism of the simulation.

B. Outlook

Addressing the first limitation of having no path planing would require the integration of more sophisticated navigation systems to enable better path planning and diversification of routes.

Overcoming the second limitation of traffic light detection would require higher resolution in simulator image quality to facilitate improved traffic light recognition and adherence.

Mitigating the third limitation of simulator would require enhancing the behavior and decision-making algorithms of

the simulated vehicles to align more closely with real-world traffic norms.

Despite having this limitations, this projects prove that purely vision-based autonomous driving can be realized successfully in simulated city environment and hopefully in real-life in the near future.

V. References

References

[1] Kaiming He et al., "Deep Residual Learning for Image Recognition" *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015

[2] Ren, Shaoqing, et al., "Faster r-cnn: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems 28*, 2015

[3] Lin, Tsung-Yi, et al., "Microsoft coco: Common objects in context."" *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 2014

[4] Andrew Howard and Mark Sandler and Grace Chu and Liang-Chieh Chen and Bo Chen and Mingxing Tan and Weijun Wang and Yukun Zhu and Ruoming Pang and Vijay Vasudevan and Quoc V. Le and Hartwig Adam,"Searching for MobileNetV3" *Proceedings of the IEEE/CVF international conference on computer vision.*, 2019