End-to-End Deep Path Planning and Automatic Emergency Braking Camera Cocoon-based Solution

Eslam Mohammed[*]
Deep Learning Researcher
Valeo, Egypt
`islam.bakr.2017@gmail.com`

Mohammed Abdou[*]
Senior Algorithms Engineer and
Deep Learning Researcher
Valeo, Egypt
`mohammed.abdou@valeo.com`

Omar Ahmed Nasr
Associate Professor,
Cairo University, Egypt
`omaranasr@cu.edu.eg`

Abstract

The autonomous driving field is progressing at a fast pace due to the revolution introduced by Deep Learning technologies in Computer Vision algorithms. To have a functioning autonomous driving vehicle, the system needs to have different functional blocks, e.g, automatic emergency braking (AEB), lane-keeping assist (LKA), active cruise control (ACC), traffic jam assist (TJA), and crash avoidance (CA). Path planning is one of the main blocks of an autonomous driving pipeline that is integrated into the system to ensure that these functionalities are working properly. Traditional path planning algorithms require difficult prerequisites: a) a well-defined map, b) sensor fusion, c) localization, and d) a controller. Unlike the traditional techniques, End-to-End deep path planning and automatic emergency braking solutions are proposed to solve the problem using a single block, instead of the complexities of the four previous blocks. In this paper, we rely only on cameras cocoon that covers 360^o around the vehicle. These cameras are installed on the four sides of the vehicle. They capture a continuous flow of images that are processed by a deep convolutional neural network. The network's output controls three vehicle's components: gas throttle, the steering wheel, and the brakes. We build our own benchmark based on the CARLA simulator due to the in-existence of benchmarks serving our proposed idea. The proposed model shows a generalization capability to differentiate between overtaking and braking, and it is considered as the first deep learning solution for both path planning and automatic emergency braking functionalities. In order to form a good basis that could help to accelerate the contribution in the field of autonomous driving, code and videos are available at `https://github.com/eslambakr/Path_Planning_using_Deeplearning`

1 Introduction

Deep learning is contributing greatly to many Automotive applications like Autonomous Driving (AD), Augmented Reality (AR). automatic emergency braking (AEB) [1], Lane-Keeping Assist (LKA) [2], Active Cruise Control (ACC) [3], Traffic Jam Assist (TJA) [4], and Crash Avoidance (CA). Forward Collision Warning (FCW) and Automatic Emergency Braking (AEB) are considered as the initial trials to integrate crash avoidance functionality [5]. In FCW, the vehicle has the ability to warn the driver that there is an object in front of the vehicle. The driver has the responsibility of taking reasonable action. However, in AEB the vehicle starts taking action through braking in case the vehicle comes close to an object in front of it. These actions are taken by the ego-vehicle based on integrating advanced sensors like Laser, Camera, and Radar. However, these vehicle's actions lead to the occurrence of many crashes, in addition to being a source of congestion because of its poorness.

[*] equal contributions

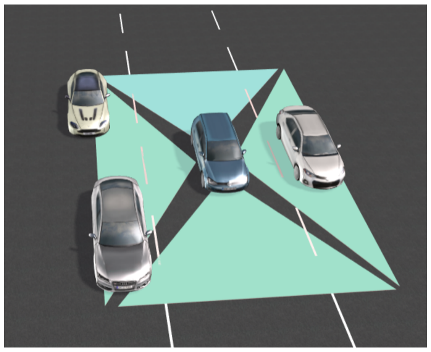

Figure 1: Camera Cocoon Vehicle Setup

Fully autonomous driving is one of the difficult problems facing the automotive applications. It is not legal yet in most of the world due to lack of trust in the technology. Big research and development efforts are curried out in order to increase the reliability of the technology, and to make it easier for the decision makers to all the technology to be implemented in a large scale. Crash avoidance functionality is considered as one of the most important features in self-driving Cars. Recently, it is partially integrated into the self-driving car system.

The path planning problem is one of the most important problems which autonomous driving cars face, as it is a safety-critical task, and needs full knowledge of the surrounding environment. During the last ten years, path planning problem was tackled using two approaches: multi-stage pipeline, and single-stage approach.

The multi-stage approach decomposes the problem into the following blocks: a) perception which is responsible for perceiving the surrounding environment, b) trajectory prediction for the surrounding objects, c) trajectory planning which compute the trajectory depends on the perception and prediction for the environment, and d) control block which is responsible for taking the good actions depending on the whole information which are gathered by the previous block. The main disadvantages of this approach are that the error is accumulated through the pipeline, this means that small errors at earlier stages in the pipeline will be propagated through other stages.

The single-stage approach, which is also called the End-To-End approach [6] [7] [8] [9], tackles the autonomous driving problem as one block instead of splitting it into many blocks. This block is responsible for generating suitable controls directly from the sensors' readings.

In this work, our contributions tackle both the path planning, and automatic emergency braking problems as follows:

- We introduce a single-stage end-to-end solution for path planning and automatic emergency braking based on Deep Learning. This solution depends on camera cocoon images covering 360^o around the vehicle, and vehicle's speed as inputs. The output of the network is vehicle's control actions: throttle, steering angle, and brake.
- Building a CARLA Simulator Benchmark to serve the installation of Camera Cocoon Sensors as shown in figure 1.
- Building a robust model with high generalization capability using different input representation.

2 Literature Review

Autonomous driving is a fast growing research domain due to the importance of the application. [2] Proposed an autonomous driving framework using deep reinforcement learning solving either Markov Decision Process (MDP) or Partially Observer Markov Decision Process (POMDP). It depends on sensor fusion for the aim of having robust readings for the surrounding environment. It controls the vehicle's manipulated parameters: throttle, steering angle by either continuous action algorithms or discrete action algorithms. [6] proposed an end-to-end behavioral cloning solution taking 3 front

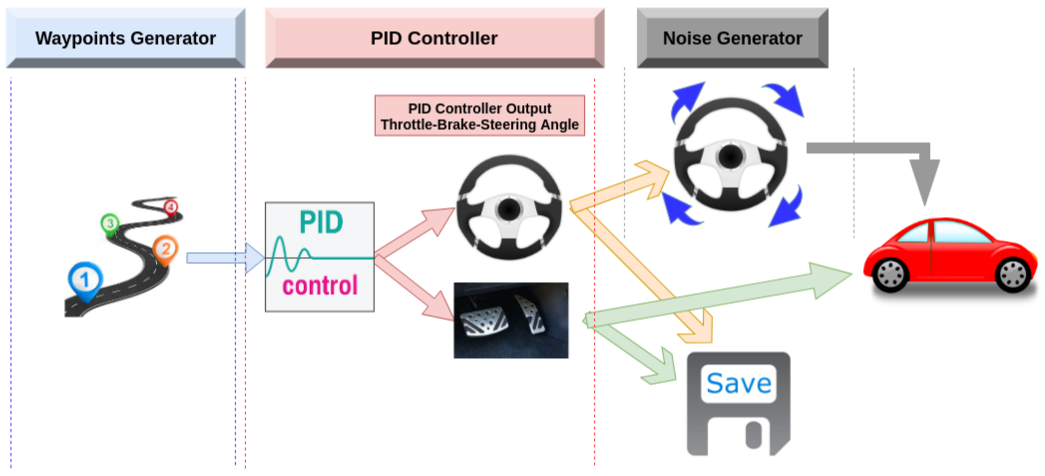

Figure 2: Data collection pipeline

camera images installed on the ego-vehicle dashboard, then applied some sort of feature extraction via convolution layers, solving a regression problem controlling the vehicle's steering angle. [9] proposed conditional imitation learning by introducing heads idea for training a deep model on straight, left, right, and follow-lane. It decomposed each task of them into heads switching between them via the control signal sent by Carla Simulator. [6] and [9] proposed a solution for lane-keeping assist functionality, not for either autonomous driving or path planning. This is because of not interactions with dynamic objects in the environment like other vehicles, pedestrians, cyclists, etc.

The single-stage approach is the end-to-end solutions that depend only on an input to the block, and outputs the control actions for the vehcile (steering, throttle, and brake). In [10], Dean Pomerleau et al. depend only on a front camera and a laser range finder image to allow the vehicle following the lane. Huazhe Xu et al., in [11], depend on the motion model of the vehicle, GPS sensor, front camera, and the previous control action to output a reinforcement learning policy taking the next control action. In [12], Simon Hecker et al. depend on TOMTOM offline saved Map, GPS, and surrounding camera views to take the best control. This means that this paper merges between the traditional way of path planning with the deep learning solution. However, this map needs to be updated frequently to avoid any change happened which is inefficient. We are proposing camera cocoon covering 360 degrees around the vehicle, and using different input representations for the captured images to achieve path planning and automatic emergency braking (AEB) functionalities through a deep neural network model.

Multistage traditional approach as in [13] and [14] is composed of 4 cascaded blocks: 1) localization block, 2) sensor fusion block, 3) waypoints generator block, and 4) control block. The localization block localizes the vehicle based on a GPS sensor in a predefined map. Sensor fusion block achieves a robust localization for the vehicle, and better environment perception for the surrounded objects. Many sensors can be fused as lidar, radar, and camera. Waypoints generator is responsible for generating waypoints ahead knowing the driving freespace so that it could slow down when there is another vehicle in front of it or it could maneuver. The control block takes the reasonable vehicle's control actions by using PID controller, or model predictive control (MPC) controller. Recently published papers are following the previous multistage approach with different combinations of the defined blocks. Some use sensor fusion between only lidar and radar, others use lidar only, others use camera as in [15] and [16], and others integrate it with lidar. Some researchers use the PID controller, others use the MPC controller, others use a hybrid controller between both of them.

Path planning requires essential parameters needed to be satisfied in order to have proper functionality. There must be a defined map for the environment, various sensor fusion like between GPS; camera; lidar; and radar, and way-points trajectory generations. The vehicle is localized depending on either the urban area or highway environment defined map and depending on GPS sensor knowing the current vehicle lane in the road. Environment perception is ensured using sensor fusion between different sources like camera, radar, and lidar achieving robust sensor readings. Camera aims to

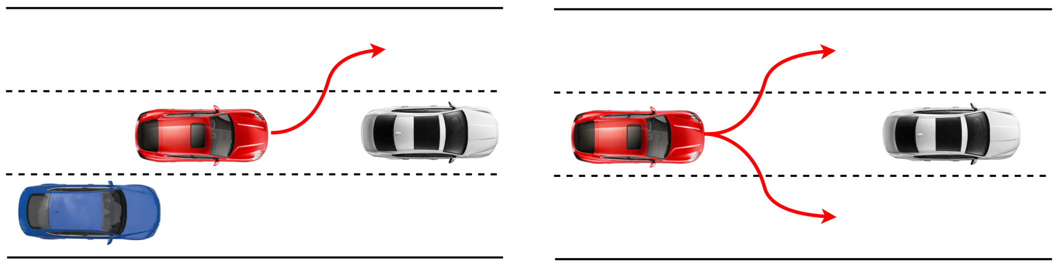
Figure 3: Path Planning Cases

classify both dynamic and static objects in the road, while radars and lidars are fused to classify objects and know the distance away from other objects. The vehicle depends on the previous to generate way-points ahead in all of the possible routes given that knowing locations of other objects. Every point represents a state, so the vehicle moves from one state to another showing the progress in the vehicle motion. An efficient finite state machine is constructed to take proper actions: keep going in the same lane, increase the vehicle speed gradually, overtake the ahead object so change the lane, decrease the speed gradually, do not change the lane due to other object is coming from the adjacent Lanes. Model predictive control (MPC) or PID controller can be used in order to control vehicle's control actions like steering angle and throttle.

The following section 3 illustrates our proposed architecture model in addition to dataset preparation, then section 4 illustrates the experimental setup and results showing different experiments with their performance. Finally, section 5 describes the main conclusion out of our proposed deep path planning model, automatic emergency braking model.

3 Methodology

This section is decomposed into three main parts: data collection phase, proposed network architecture and different input representations.

3.1 Data Collection

The data collection pipeline consists of three main blocks as shown in Figure 2. The first block is the waypoints generator which is considered as the most important block in the data collection phase. Waypoints are generated based on the well-defined map using CARLA simulator [17], then a post-processing phase is applied on these waypoints to act as the traditional ground truth for path planning functionality. The second block is a PID controller which is composed of proportional, integral and derivative components which are tuned to achieve a smooth performance for the vehicle motion following the previously generated waypoints by controlling three vehicle actions: throttle, steering angle and brake. The third block is the noise generator which is inspired by CARLA simulator [17] in which a noise is injected into the steering during data collection. Namely, at random points in time we added a perturbation to the steering angle provided by the PID controller. The perturbation is a triangular impulse: it increases linearly, reaches a maximal value, and then linearly declines. This simulates gradual drift from the desired trajectory, similar to what might happen with a poorly trained controller. The triangular impulse is parametrized by its starting time t_0, duration $\tau \in R^+$, $\sigma \in \{$-1,+1$\}$, and intensity $\gamma \in R^+$ as in the following equation:

$$S_{perturb}(t) = \sigma\gamma \max\left(0, \left(1 - \left|\frac{2(t-t_o)}{\tau} - 1\right|\right)\right) \tag{1}$$

Our main aim from this pipeline is to generate training data: images using camera cocoon setup covering 360^o, and labels of the vehicle's control action.

We collected 1500 episodes which contain 440k data frames, each data frame consists of 4 images from the forward camera, the right camera, the left camera and the backward camera with its corresponding measurements which consist of throttle, brake, steering percentage and forward speed

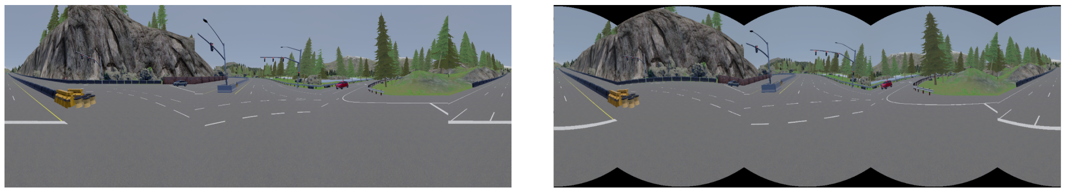
Figure 4: Normal stitching vs. Equirectangular projected Image

of the car. The episode starts from a random location on the map and finished when our vehicle reaches the desired destination, and for each new episode, we randomize the start and the destination position, number of other vehicles, the type of the vehicles and even their colors. Then the dataset is split to training data set and validation data set as follows 80% and 20% respectively. The data is collected covering various and different scenarios, our vehicle is moving and the other vehicles are static. In order to ensure the robustness for our model, we collect data while other vehicles are in static and dynamic states. Our main goal is to have an end-to-end deep model that is able to predict the vehicle's control action properly like overtaking the ahead vehicle safely, resuming in the current lane, and braking in case of absence of reasonable trajectory. That's why the collected data is taking safety aspects into consideration, so we make sure if the other lane is empty or not as shown in figure 3.

3.2 Network Architecture

Convolution neural network is used as the feature extractor followed by fully connected layers to map the input which consists of raw pixels from camera cocoon and the vehicle speed in the current timestamp to the three actions directly, throttle, steering angle and brake. Figure 6 in the appendix shows our proposed model architecture in which the input image path uses eight convolution layers followed by two fully connected layers. The input vehicle's speed go through two fully connected layers, then the last two fully connected layers which come from the image's path and the speed's path are concatenated with each other then the concatenated layer is followed by three fully connected layers then the output layer. This problem is considered as regression on the continuous control actions, Adam optimizer is used with decayed learning rate as in the following equation 2 starting from 10^{-4}, and decaying rate equals to 0.96. Figure 7 in the appendix shows our proposed model architecture for inputting the four images separated, by duplicating the image path four times each input image uses separate image path exactly as shown in figure 6. The CNNs in the image path could share it's parameters with each other or to be fully separated.

$$decayed\_learning\_rate = learning\_rate * decay\_rate^{epoch\_number} \tag{2}$$

3.3 Input Representations

3.3.1 Four Separated Images

Four separated images representation depends on the camera cocoon vehicle's installation, it simply takes the four separated images which are front, right, left, and rear views as input to a DNN which outputs the vehicle's control actions. This representation is used in conducting two experiments: the first one depends on sharing the same parameters of the features extraction part of the four images, and the second one depends on not sharing their parameters. Inspired by [6], four images are input together with sharing parameters of their feature extraction part, each image is stacked with the previous three images from the same camera. These images are fed to our model to ensure having well perception for the surrounded environments and generalizing well in unseen situations.

3.3.2 Equirectangular Projection

Equirectangular projection is a cylindrical equidistant projection which is used in map creation by Marinus of Tyre. In equirectangular panoramic image, all verticals remain vertical, and the horizon becomes a straight line across the middle of the image. The poles (Zenith, Nadir) are located at the top and bottom edge and are stretched to the entire width of the image. To generate the equirectangular

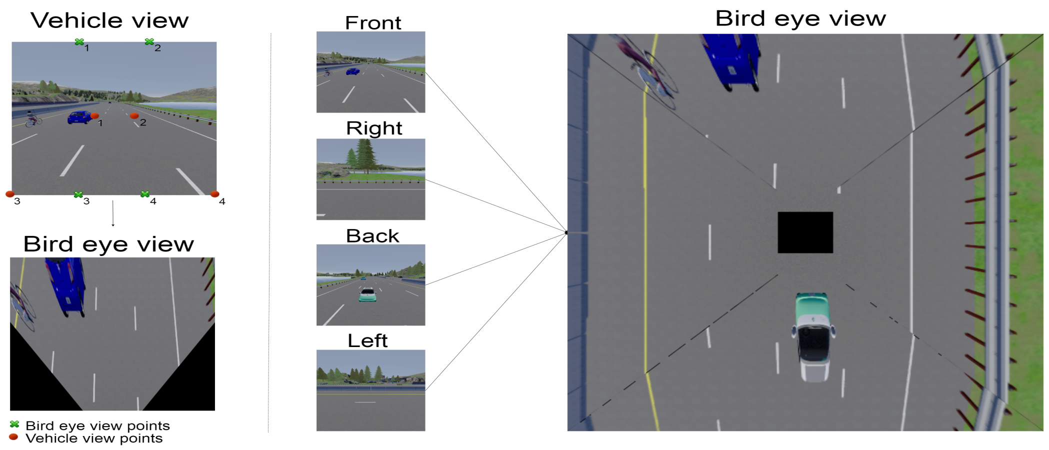

Figure 5: Warping vehicle view image into bird's eye view

projection we use the cubic projection in which both of them are used for mapping the surface of a sphere to flat images. By Imagining the surrounding view around the vehicle is the sphere, we need to convert this sphere to a flat image. To achieve that, we need first to convert the sphere to the cubic format, then convert the cubic format to the equirectangular projection. Cubic format consist of four cube faces cover front, right, back and left, one the zenith and one the nadir, each of them having 90^ox90^o field of view as shown in Figure 8 in the appendix. Our cameras setup on the vehicle captures four images representing the four cube faces and replacing the zenith and the nadir by black images as in this application there is no need to cover the upper and the lower area around the vehicle. Using Transformation equation, we could translate the cubic format to equirectangular projection. The only difference among our generated image and the original equirectangular projection is that we clip the top and bottom parts as there is no need of them in our application.

3.3.3 Bird's Eye View Projection

Bird's eye view is widely used in autonomous vehicles but only in auto parking systems. In this work, we extend it's usage to propose an end-to-end driving system using deep learning. Bird's-Eye representation shows a near range surrounding view around the vehicle so it is challenging to use it in highway lane over taking task which requires a long range information to enable the autonomous system taking the right decision properly. The key solution which is used to make the projection suitable for our task is to choose the reasonable ROI from the normal camera (vehicle) view to be projected on the bird's eye view. To warp the vehicle view to bird's eye view, we just need some locations from the vehicle view and the respective locations of the desired perspective (bird's eye view). These location coordinates are called the source points and destination points respectively, and they have been chosen manually to achieve the best projection which aligns with the highway challenge. This can be done by taking the advantage of the lanes as shown in the left part of figure 5. From the four cameras, a bird's eye view image is generated as shown in the right part of figure 5. There is no need to use RANSAC [18] to determine the homography as we have chosen the four points manually and we are sure they are correct, directly calculates the 3×3 matrix of a perspective transform as in the following equation 3:

$$\begin{bmatrix} t_i x_i^{'} \\ t_i y_i^{'} \\ t_i \end{bmatrix} = map\_matrix. \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3}$$

$$dst(i) = (x_i^{'}, y_i^{'}), src(i) = (x_i, y_i), i = 0, 1, 2, 3$$

4 Experiment

In this section, we will show the non existence of a suitable real life datasets suitable for the applications in the paper. We then discuss input representations for the network. We illustrate our proposed benchmark for path planning. After that, our experimental results are showed comparing between various representations.

4.1 Real Datasets

Testing on real data will be beneficial, but unfortunately we did not find any relevant open source data. KITTI [19], Cityscapes [20], Comma.ai, Berkeley DeepDrive and Baidu Apolloscapes datasets were collected using forward camera only. Although there are other datasets which cover the surrounding view as Oxford's Robotic Car [21], Argoverse [22] and nuScenes [23] datasets, but those data don't contain vehicle's action labels like steering angle and throttle. The control data is important to train the network. Drive360 [12] contains vehicle's action labels and the surrounding view around the vehicle, but it targets another application and hasn't been captured on highways, so we can not compare our work with existing approaches experimentally as no one targeting our application using deep learning. We aim to expand our work by collecting a real data in the future.

4.2 Experimental Setup

A major contribution in this paper is in the different representations for the input: baseline representation, four separated images representation, and equirectangular projection representation. In all of our representations, we stack the current image with the previous 3 images to encode the temporal information and train the model for 30 epochs with decayed learning rate as in equation 2 starting from 10^{-4}, and the decaying rate equals to 0.96. The baseline model has **11 million** parameters, four separated images with parameter sharing model has **41 million** parameters, however without parameter sharing has **45 million**, the equirectangular projection model has **13 million** parameters, and finally, the bird's eye view model has **25 million** parameters.

4.3 Path Planning Benchmark

Targeting autonomous driving tasks using deep learning, a robust benchmark is required to be able to judge how good the agent is doing. Evaluating the agent is not an easy task as it requires covering a large number of corner cases, and new situations in which the agent has never seen before. These tasks have not been tackled using deep learning as path planning, lane overtaking and automatic emergency braking, so we have to create our benchmark. The existing benchmarks do not meet our requirements, however, in [17] they focus on the success rate for turning right and left, the percentage of lane crossing by the agent and the collision percentage.

Table 1: Training Town Benchmark Results

Experiment	Scenarios	Collisions	AEB Stops	Distance
Baseline Experiment	4	2	13	36%
Equirectangular Projection	7	**0**	12	48.67%
Four Images with Sharing	6	1	12	39%
Four Images without Sharing	4	1	14	30%
bird's Eye View	**8**	3	**8**	**53.985%**

Our benchmark is built using CARLA simulator [17], it consists of 19 different scenarios which cover the whole high way town in CARLA covering tough situations like an inverted car, crashes on the road, cars stop horizontally across two lanes of the highway and upside-down cars as shown in the appendix. Our benchmark depends also on 2 towns, CARLA town 4 whose distance equals **2614 meters** is used as the training town, however, CARLA town 5 whose distance equals **1103 meters** is used as the testing town. These towns are chosen to simulate highway roads. We evaluate our model on another town with different scenarios and situations for the aim of testing the generalization capability of our model on unseen situations. In each scenario, the agent starts from a fixed position and aims to reach a specific destination without collision with other vehicles, fences on both sides of the road or any other objects.

4.4 Experimental Results

The evaluation process mainly depends on three different measurements to differentiate between the different models performance. These measurements metrics are as follows:

- how many scenarios the agent succeed to reach the desired destination,
- how many scenarios the agent collides with other vehicles or any other object in the environment,
- how many times the agent prefers stopping rather than changing its lane, as it is a safety critical task. This is why automatic emergency braking is ensured by our proposed model,
- distance percentage covered successfully with respect to the total distance of the training and testing towns.

Evaluation metrics are applied on both training, and testing towns to test the generalization capabilities of our proposed models

Table 2: Testing Town Benchmark Results

Experiment	Scenarios	Collisions	AEB Stops	Distance
Baseline Experiment	4	**0**	8	54.6%
Equirectangular Projection	7	2	**3**	71.4%
Four Images with Sharing	6	1	5	68.5%
Four Images without Sharing	4	2	6	52%
bird's Eye View	**8**	1	**3**	**74.81%**

Table 1 shows training town benchmark results. **The baseline** experiment fails due to using a single camera only installed in front of the ego-vehicle. Logically, it fails in path planning functionality because it does not take the surrounded environment into consideration. In addition to that, **four images** stitched together **without sharing** their features also fails in achieving good results. However, our proposed architecture achieves the best performance using either the **equirectangular projection** or the **four images with shared parameters** in their feature extraction part and outperform other models. Although these images came from four different cameras, **sharing** the extracted feature from them enhances the model performance enforcing the model to understand various views and relate them to each other. In addition, equirectangular projection succeeds in extract spatial information from the stitched image differentiating between the different parts of the stitched image to understand the importance of each part.

Table 2 aligns with the the training town benchmark results and shows the testing town benchmark results. **Baseline** experiment and **four images** stitched together **without sharing** their features fail in achieving good results. However, our proposed methods, **equirectangular projection** and the **four images with shared parameters** in their feature extraction part, outperform other model's performance and overgeneralize over different town used in the training phase.

The emergency stopping scenario means that the vehicle is facing an obstacle and it has the chance to change the lane but the agent prefer to not change the lane, so the agent is stuck and couldn't finish the scenario. That's why in table 2 the baseline achieved zero collisions. However, we consider it the worst approach as the vehicle didn't even reach to the position where the others collide with an obstacle. Therefore, we added the traveled distance to the results as we can't rely only on the number of the succeeded scenarios.

5 Conclusion

Path planning and automatic emergency braking are considered as two of the main functionalities required to have a fully autonomous driving vehicle. In this paper, we have introduced a single stage path planning and automatic emergency braking modules based on deep learning. The algorithms relies only on a camera cocoon installed covering 360^o around the vehicle, instead of having multiple sensors. The input of the neural network is the images of the vehicle surroundings, and the output is the control actions for the car (gas throttle value, the steering wheel angle, and the brakes). Different techniques to combine the images from the camera were studied. We built our benchmark based on CARLA simulator covering difficult and various scenarios or situations that the vehicle may experience. We have shown that using a bird's eye view, and sharing the same parameters of the features extraction part of the four images in the network gives superior performance compared to using a single image or not sharing the features. Moreover, we have shown that the network has good generalization capabilities in different scenarios.

References

[1] O Garcia-Bedoya, S Hirota, and JV Ferreira. Control system design for an automatic emergency braking system in a sedan vehicle. In *2019 2nd Latin American Conference on Intelligent Transportation Systems (ITS LATAM)*, pages 1–6. IEEE, 2019.

[2] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. End-to-end deep reinforcement learning for lane keeping assist. *arXiv preprint arXiv:1612.04340*, 2016.

[3] Jianbo Lu, Kwaku O Prakah-Asante, and Dimitar Petrov Filev. Smart adaptive cruise control, November 10 2015. US Patent 9,180,890.

[4] Rinku Patel, Christopher Melgar, Andrew Niedert, and Phil Lenius. Enhanced traffic jam assist, May 30 2019. US Patent App. 15/826,973.

[5] Yuji Kodama and Koji Nakatani. Collision avoidance system, November 27 2018. US Patent App. 10/140,867.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. pages 1–9, Apr 2016.

[7] Hesham M. Eraqi, Mohamed N. Moustafa, and Jens Honer. End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. pages 1–8, Oct 2017.

[8] Zhilu Chen and Xinming Huang. End-to-end learning for lane keeping of self-driving cars. June 2017.

[9] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. Oct 2017.

[10] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[11] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182, 2017.

[12] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 435–453, 2018.

[13] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[14] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Ferreira Reis Jesus, Rodrigo Ferreira Berriel, Thiago Meireles Paixão, Filipe Mutz, et al. Self-driving cars: A survey. *arXiv preprint arXiv:1901.04407*, 2019.

[15] Zhengyuan Yang, Yixuan Zhang, Jerry Yu, Junjie Cai, and Jiebo Luo. End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2289–2294. IEEE, 2018.

[16] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE international conference on computer vision*, pages 2942–2950, 2017.

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[18] Ziv Yaniv. Random sample consensus (ransac) algorithm, a generic implementation. *Insight Journal*, 2010.

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[20] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[21] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[22] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[23] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

6 Appendix I

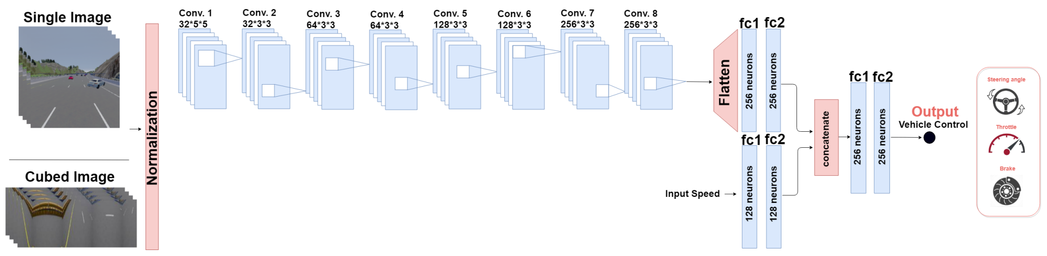

Figure 6: Deep Path Planning and Automatic Emergency Braking Architecture using Front camera or camera cocoon images cubed in one image

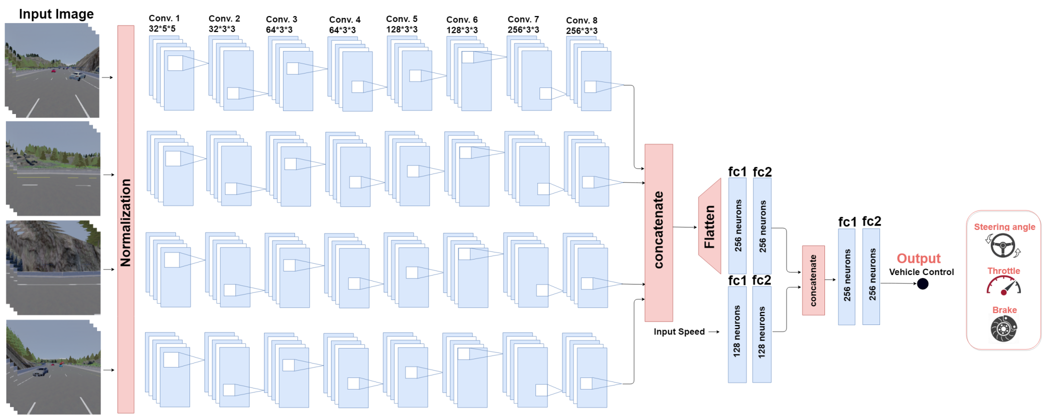

Figure 7: Deep Path Planning and Automatic Emergency Braking Architecture using camera cocoon images. Feature Extraction part may be shared or not.

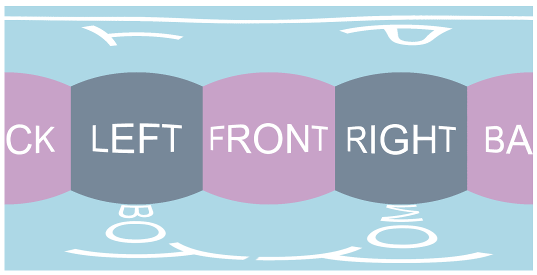

Figure 8: Equirectanguler Projection Technique

This video shows the differences among different representations, the first one shows the difference among equirectangular and normal stitching, and the second differentiate among the three projections' types, the equirectangular, Normal stitching and the bird's eye view. (https://drive.google.com/open?id=1ACGC1yYREdSM-aRq6Xu-2iXZbEDObKhx).

To clarify the results, collision and stopping scenarios, this video (https://drive.google.com/open?id=18mOoHr1C25ZPvtSskqeshYJXPefpJ0xG) shows samples from the benchmark results which show samples for the succeeded, collision and emergency stopping scenarios. The video shows the scenario in-which the majority of the proposed input representation fails to pass it as a small part of the car occupies another lane of the road.

Our Benchmark cover various, difficult, and different situations as shown in the following screenshots from CARLA Simulator: