ICIC Express Letters
Part B: Applications
Volume **13**, Number **9**, September **2022**

ICIC International ©2022 ISSN 2185-2766
pp. **975–982**

SINGLE CAMERA DATA AUGMENTATION IN END-TO-END DEEP LEARNING SIMULATED SELF-DRIVING CAR

Muhammad Idham Ananta Timur*, Jazi Eko Istiyanto
Andi Dharmawan and Beatrice Paulina Setiadi

Department of Computer Science and Electronics
Universitas Gadjah Mada
Sekip Utara, BLS 21, Yogyakarta 55281, Indonesia
{jazi; andi_dharmawan}@ugm.ac.id; beatrice.p@mail.ugm.ac.id
*Corresponding author: idham@ugm.ac.id

Received November 2021; accepted February 2022

Abstract. *Developing a self-driving car is a daunting task. Usually there are multiple steps involved in the learning pipeline to generate a feasible model. This research offers an alternative approach using end-to-end deep learning with vast data generated from a simulated environment. The dataset, combined with data augmentation, is expected to contain enough features to be learned. The dataset for training is taken by manually driving the car, using the record feature in the simulator to store the dataset. It consists of vehicle's driving behaviour labels and images which are taken from a single camera mounted on the car's dashboard. Convolutional Neural Network (CNN) is used to process the copy and labels of the dataset while training the model. The result of this research is a simulation that is able to steer the car, in which the trained model makes the predictions for the steering angle based on image input from the camera. The accuracy of the trained model is measured through the RMSE calculation, which results in a value of 0.178. We also evaluate the model according to their time and distance in autonomous driving testing from different start points on the map.*

Keywords: Self-driving car, Data augmentation, End-to-end learning, Simulated environment

1. Introduction.

Today, the world is in the development of autonomous car or self-driving car. This vehicle is trained to make decisions related to driving using artificial intelligence, in which a computer has a role to take over and fully control the steering [1]. One of the methods to develop self-driving cars is through deep learning. Deep learning is a type of machine learning that mimics the neurons from the neural network in the human brain. This deep learning method can learn patterns in images presented as the training data rather than requiring human operators to determine the patterns that the machine must look for in those images [2].

In this research, rather than doing a complex multiple steps learning, we adopt an end-to-end deep learning approach to simply develop an autonomous model that is able to keep the car on the lane based on a single front camera and several car's control attributes. This approach does not have a manual feature engineering to recognize object like a lane, a car or else. Instead, an end-to-end manual driving around the map is used to collect the dataset. It is expected that the data contain enough features to be learnt using deep learning method.

In many real-world scenarios, self-driving cars are equipped with many sensors such as radar, LiDAR, infrared, and cameras [3]. Each sensor has its own notable shortcomings. LiDAR, for example, has lower input data resolution compared to camera [4], especially

DOI: 10.24507/icicelb.13.09.975

from distance, while cameras might have exposure problems in different time of driving. Using many sensors will require specific calibration, alignment, and to some extend cross-modality augmentation, which are not suitable for our universal end-to-end learning approach. Thus, in this paper we focus on using single camera to explore data augmentation techniques to tackle those exposure problems.

This car is trained in a simulated environment, AirSim, which is a simulator designed for autonomous vehicles (e.g., drones, and cars), built on Unreal Engine with open-source, cross-platform, and supports hardware-in-loop [5]. The purpose of developing a self-driving car in a simulator is to reduce the risk of direct testing on the highway, reducing expenses, and accelerate the development of autonomous vehicles [6]. Training a model using end-to-end deep learning method will require a lot of data, so using a simulator is suitable for training the model because simulator has the potential to produce unlimited amounts of data.

Huval et al. [7] evaluated a variety of deep learning techniques to computer vision problems in highway perception scenarios empirically. Empirical evaluation referred to the assessment of combined deep learning methods with computer vision through observation in experiments. CNN method was applied to detecting paths and objects on the highway in real time. The result of this research showed that the implementation of CNN in deep learning method provided the best solution for the development of autonomous driving which can detect lane on highway and objects or vehicles around it. Chen et al. [8] conducted a study on autonomous driving using the paradigm of the direct perception approach. The demonstration of the direct perception approach was done by training a deep convolutional neural network from a recorded dataset of 12 hours human driver in The Open Racing Car Simulator (TORCS) video game and compared it with a dataset from KITTI. The result showed that the perception approach directly yields a model that can work well to control a car in a variety of virtual environments and roads in the real world. Bojarski et al. [9] conducted a study of Convolutional Neural Network (CNN) to map the pixels of an image input from a camera to be able to produce steering commands directly. The end-to-end approach is used to prove that with a minimum of training data, the system can learn to run on local roads without road markings and highways. The results showed that CNN could learn lane-following tasks without manual parsing for path detection or road signs, semantic abstractions, path planning, and control. Chi and Mu [10] researched a visual-based model which can map a raw data input in the form of image input and conditions of the vehicle become the steering angle output using a deep network for autonomous driving purpose. The results of this research showed that the deep network which processed the combination of spatial information (images) and the condition of the vehicle could produce an excellent performance in predicting the steering angle for the development of autonomous driving. Recently, both CNN and long short-term memory (LSTM) are used to build end-to-end deep learning model for autonomous driving in high-speed environments [11] in a way that is similar to our approach. However, in this paper we explore more about data augmentation and model evaluation.

The rest of the paper is structured as follows: Section 2 addresses simulated environments configuration and learning design, Section 3 discusses data augmentation techniques and training model, Section 4 describes model evaluation, and Section 5 concludes the paper.

2. **Simulated Environments and Learning Design.**

The research was conducted by training and testing an end-to-end deep learning model to control a car automatically based on a dataset collected by driving on the Neighborhood Environment from AirSim simulator. The steering angle prediction produced by the trained model was meant to control the car to run on the Neighborhood Environment, in which a camera mounted on the car's dashboard is used to get visual inputs (images). Neighborhood Environment

Figure 1. Neighborhood Environment of AirSim simulator

is one of the environments built on the AirSim simulator that resembles a residence. No other vehicles are running, pedestrians, or road signs on this Neighborhood Environment. The Neighborhood Environment is shown in Figure 1.

Overall, the design of this system consists of two processes: training, and testing. The training process began with collecting and preparing the dataset by manually driving a car to obtain image data and car's condition labels. Image data collected from manually operating the vehicle was augmented and extracted based on the Region of Interest (ROI). Obtaining ROI means to make certain areas of the image needed to train the model. The training process used Convolutional Neural Network (CNN) method to produce the model. This model was prepared by using those pre-processed images and the car's condition labels as a dataset. Afterwards, in the testing process, the system would use the best model produced by the training process and read current image input from a camera mounted on the car. That image input would also be extracted based on the ROI. The best model was tested by placing the vehicle at specific starting points fifteen times for each position. We use Python as control interface to the AirSim system which referred to Autonomous Driving Cookbook [12]. The block diagram of the overall system design is shown in Figure 2.

The training dataset was collected by driving a car manually for nine times. AirSim simulator has a record feature which allows the user to save a dataset, consisting of the vehicle's condition labels and recorded images. The example of label data obtained from the record feature is shown in Figure 3. The dataset was divided into two types of driving strategies: standard, and swerve. Regular driving strategy maintained the steering angle close to zero value which made the car mostly go straight on the road. Swerve driving strategy made the car almost always oscillated across the street. The model depends practically entirely on the dataset to provide the information needed for training. Therefore, to overcome any sharp turns that the model might encounter and to give it the ability to correct itself if it starts to go off the road, then the model needs to be trained with such driving examples. The images and labels were combined into compressed data files in h5 format file. The h5 format is suitable because it supports the processing of many datasets without having to read them all at once into the memory and it can work together with Keras. The prepared and compressed dataset consisted of train, validation, and test data with the ratio of 7 : 2 : 1. Train data is a sample of the dataset used to fit the model.

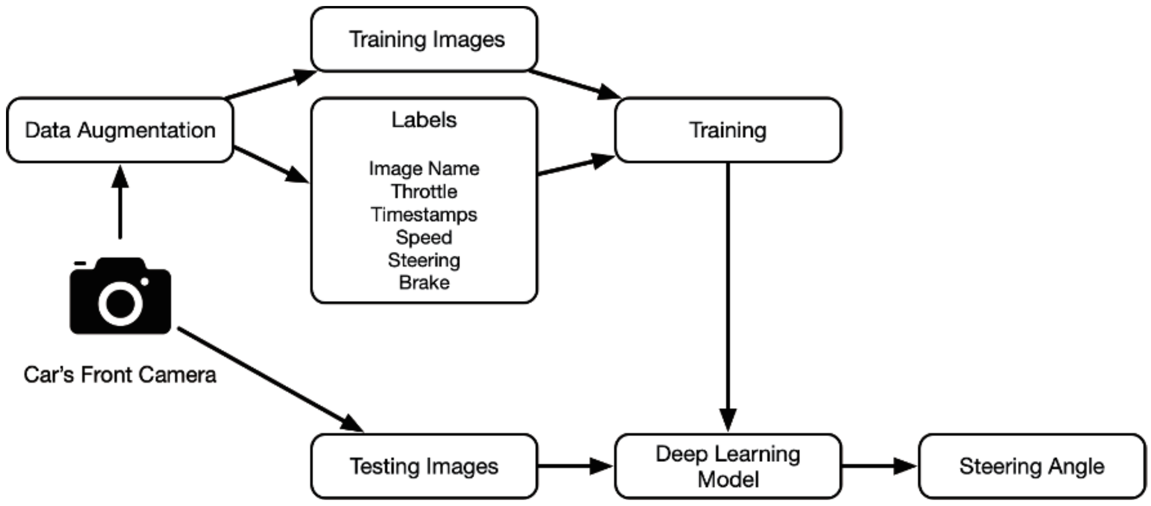
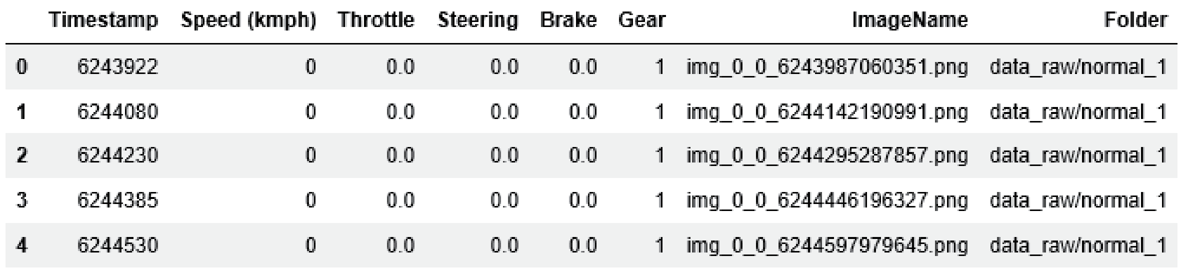


Figure 2. End-to-end deep learning design



	Timestamp	Speed (kmph)	Throttle	Steering	Brake	Gear	ImageName	Folder
0	6243922	0	0.0	0.0	0.0	1	img_0_0_6243987060351.png	data_raw/normal_1
1	6244080	0	0.0	0.0	0.0	1	img_0_0_6244142190991.png	data_raw/normal_1
2	6244230	0	0.0	0.0	0.0	1	img_0_0_6244295287857.png	data_raw/normal_1
3	6244385	0	0.0	0.0	0.0	1	img_0_0_6244446196327.png	data_raw/normal_1
4	6244530	0	0.0	0.0	0.0	1	img_0_0_6244597979645.png	data_raw/normal_1

Figure 3. Data label example

Validation data is a sample of the dataset used to evaluate the model during the training process, but it is not used for the model to learn from it. Test data is a sample of the dataset used to provide a final model evaluation.

3. **Data Augmentation and Training Model.** The compressed dataset was augmented to enrich as much data as it is possible to train the deep learning model and avoid overfitting. It is a condition when the model is only focused on specific training data. So, the model is unable to predict well when given other similar data. The data augmentation was done by horizontally flipping the image data and modifying the image brightness level to 40%, so they produced new image data. We use the concept of data generator provided by Keras to perform this data augmentation. The visualization of data augmentation is shown in Figure 4.

An only specific part of the image data is used for the training process. Therefore, the images were focused on the Region of Interest (ROI). Extracting this ROI reduced training time and the amount of data needed to train the model. It would also prevent the model from getting confused by focusing on irrelevant features in the Neighborhood Environment (e.g., tree, house, and power line). We used the concept of Keras data generator to extract the ROI of all images.

CNN architecture in this research used a standard combination of convolutional layer and max-pooling layer to process images. After early trials with various architectures, we decide to utilize three convolutional layers used with 16, 32, 32 filters, and 3×3 convolutional window. This filter or also called the convolution kernel refers to an operator which applies the entire images in a certain way to changing the information encoded in pixels. The batch size of images trained by CNN was set on 32. Image features were combined with the input layer, which contained the previous state of the car and the

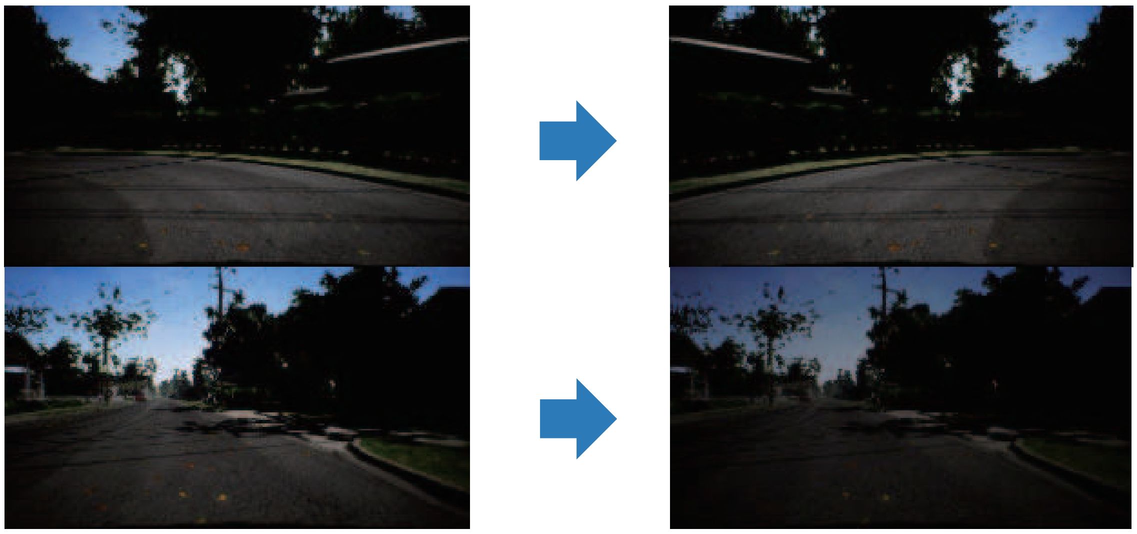


FIGURE 4. Data augmentation flipped (top) and brightness level (bottom)

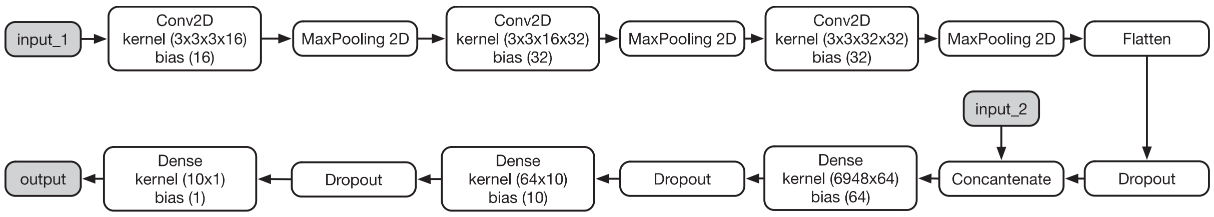

input_1
Conv2D kernel (3x3x3x16) bias (16)
MaxPooling 2D
Conv2D kernel (3x3x16x32) bias (32)
MaxPooling 2D
Conv2D kernel (3x3x32x32) bias (32)
MaxPooling 2D
Flatten
input_2
Dropout
Concantenate
Dense kernel (6948x64) bias (64)
Dropout
Dense kernel (64x10) bias (10)
Dropout
Dense kernel (10x1) bias (1)
output

FIGURE 5. CNN architecture

labels of steering wheel angle. This combination then passed through two fully connected layers which connected with 64 and 10 hidden layers. The activation function used in this network architecture was ReLU, but the last layer of the network was a single neuron without activation. The output of this network was predicted steering angles in floating-point numbers. CNN architecture is shown in Figure 5.

The generated model from the training process was evaluated to measure its accuracy by calculating the RMSE value. Root Mean Square Errors (RMSE) is a method used as an evaluation of a forecast by measuring the accuracy of the generated model. RMSE represents the square root of the second sample moment of the difference between the predicted value and the observed value or the average square of this difference. RMSE calculates the average of the squared errors summation [13]. We used 150 test data samples to evaluate the model. RMSE calculation produced a value of 0.177958. This value meant that the trained model had high accuracy. A low cost of RMSE implies that the variation of predicted values is close to the variety of actual values.

4. **Model Evaluation.**

The best-trained model was used in the testing process. The present input of images was obtained from a single camera mounted on the car's dashboard, and then these images were segmented based on the ROI as mentioned before. The concept of how the testing process works is by reading the input data of images and the present state of the car presented as labels. Then the model will make steering angle predictions based on the image data input. In this testing process, the vehicle was set at a constant speed, around 5 m/s. The model testing took place in four different starting points. Points A, B, C, and D, as shown in Figure 6, are the starting points used in the testing process.

The car was placed at these points with fifteen trials for each position (60 times model testing in total). We modified the environment's setting in the settings.json file, which

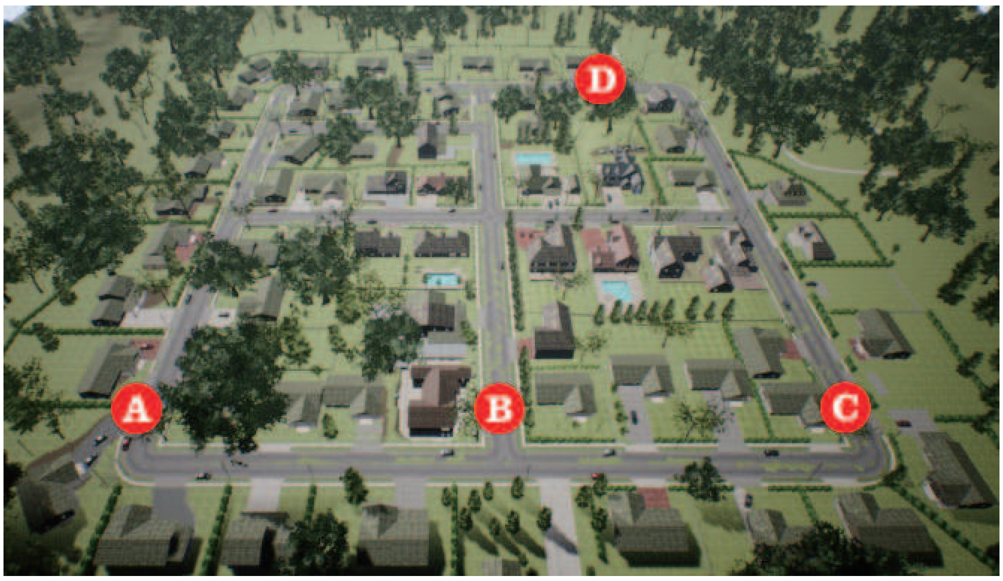


FIGURE 6. Starting points for model testing

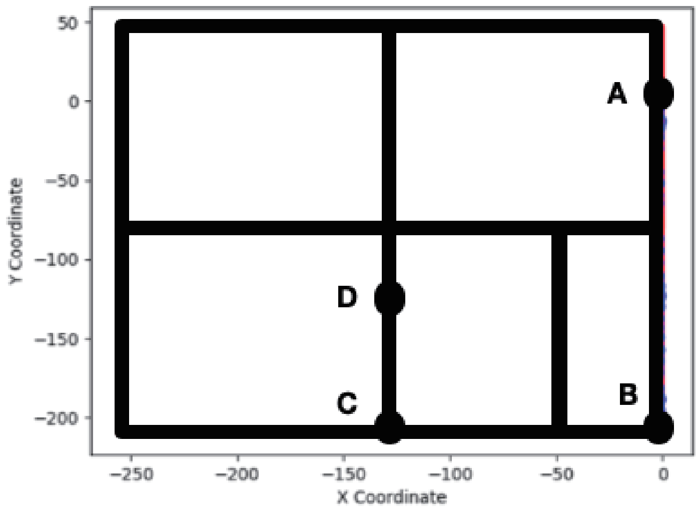


FIGURE 7. Track for model testing

refers to AirSim documentation [14]. This modification aims to manage initial position of the car and the type of vehicle. During the testing process, the X and Y coordinates of the vehicle's location will be saved as text files. We evaluated the testing process by making statistical data related to how long the car can stay on the road and how far the car can go from each starting point. These scenarios were measured as long as the vehicle does not collide with other objects in the Neighborhood Environment. When the car starts to go off the road or hit with other objects, and it has no potential to correct itself, the vehicle will return to the starting point. Every time the car returns to its starting point, it will be counted as a one-time test or trial. The statistic data of time and distance were used to observe the performance of the trained model.

We plotted the car's position during model testing and the lane of Neighbourhood Environment road. Figure 7 represents the plotting of the 4th test in starting point D. The track represents the plot of Neighborhood Environment road, whereas the points represent the coordinates of the car's position during model testing. As can be seen in Figure 7, the vehicle was able to run automatically through three intersections until it finally stopped. We mentioned before that the coordinates (X and Y axes) of the car's position were saved as text files. Therefore, the initial and final coordinates of the car's location are recorded and can be used for plotting.

Point A is the initial position of the car with a coordinate of $(0, 0)$, whereas point D is the final position with a coordinate of $(-125.32, -105.25)$. Points B and C are the known coordinate of the Neighborhood Environment road, with the values of $(0, -209.32)$ and $(-127.76, -209.32)$. Using those coordinates, we obtained the distance measurements of the model testing from each starting point, as shown in Table 1. During model testing, we also recorded how long the car was able to stay on the road using a timer. The time records of the model testing from each point are shown in Table 1. Based on it, we can see that the highest average value of how long the car can stay on the road is in starting point C. The time records show how long the car can stay on the road during model testing varied for each trial. We can conclude that the trained model has pretty good performance if we consider how limited the training dataset is. Furthermore, neighborhood environment has many variations of an intersection, which make it more difficult for the car to make sharp turns during testing and increase the risk of going off the road.

Table 1. Distance measurements (in coordinate unit)

		Distance (coordinate unit)				Duration (seconds)			
		A	B	C	D	A	B	C	D
Testing trials	1	272.6	715.18	166.54	593.37	101	267	82	180
	2	55.38	536.59	165.25	457.79	31	195	84	185
	3	341	116.25	87.47	412.4	136	63	53	138
	4	61.19	178.31	82.26	441.15	35	83	53	200
	5	55.01	173.22	82.64	200.15	34	81	49	101
	6	42.81	546.55	302.65	32.06	31	196	122	17
	7	55.36	607.55	165.17	34.72	30	248	84	20
	8	55.22	554.53	81.92	330.01	31	202	48	84
	9	55.79	174.46	909.57	31.77	32	81	360	17
	10	55.06	175.14	1093.11	530.6	31	83	417	222
	11	56.22	175.39	165.68	31.83	31	82	83	15
	12	41.88	220.58	167.13	443.77	25	94	84	181
	13	58.36	116.25	81.61	183.09	32	64	48	85
	14	56.02	174.31	305.08	33.69	31	81	120	15
	15	337.68	174.4	983.69	200.28	133	81	360	93
Average		139.87	344.53	367.33	301.8	49.6	126.73	136.47	103.53

5. **Conclusions.**

An end-to-end deep learning approach from a single camera shows a very promising model of a self-driving car. The data augmentation and some tuning on training parameters make it more efficient. We applied a convolutional neural network to processing the dataset (images and labels). The model's RMSE is 0.177958. The trained model was able to predict the car's steering angles with a relatively small amount of dataset. However, different environment might need a different strategy. The simulation can become a head-start before testing the vehicle in the real world. In this research, we created an end-to-end deep learning model which was trained to predict steering angles for a self-driving car based on image input from the single front-faced camera and the car's last known driving behaviour.

For further research, we can consider the addition of dataset for training the model with more driving strategies. We can also combine specific learning algorithms, such as deep end-to-end learning with reinforcement learning. Therefore, the car will not only be trained from a given dataset but also trained during testing.

Acknowledgment. This work is partially supported by Ministry of Education and Research of Republic Indonesia. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] M. Maurer, J. C. Gerdes, B. Lenz and H. Winner (eds.), *Autonomous Driving: Technical, Legal and Social Aspects*, Springer-Verlag, Berlin Heidelberg, 2016.

[2] R. Shanmugamani, *Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*, Packt Publishing Ltd., 2018.

[3] Y. Lai, Y. Chen and J. Perng, Sensor fusion of camera and MMW radar based on machine learning for vehicles, *International Journal of Innovative Computing, Information and Control*, vol.18, no.1, pp.271-287, 2022.

[4] Y. Li and J. Ibanez-Guzman, Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems, *arXiv Preprint*, arXiv: 2004.08467, 2020.

[5] S. Shah, D. Dey, C. Lovett and A. Kapoor, AirSim: High-fidelity visual and physical simulation for autonomous vehicles, *ArXiv170505065 Cs*, http://arxiv.org/abs/1705.05065, Accessed on Apr. 08, 2020.

[6] S. Yao, J. Zhang, Z. Hu, Y. Wang and X. Zhou, Autonomous-driving vehicle test technology based on virtual reality, *Journal of Engineering*, vol.2018, no.16, pp.1768-1771, 2018.

[7] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates and A. Y. Ng, An empirical evaluation of deep learning on highway driving, *CoRR*, vol.abs/1504.01716, 2015.

[8] C. Chen, A. Seff, A. Kornhauser and J. Xiao, DeepDriving: Learning affordance for direct perception in autonomous driving, *IEEE International Conference on Computer Vision (ICCV)*, Santiago, pp.2722-2730, 2015.

[9] M. Bojarski et al., End to end learning for self-driving cars, *ArXiv160407316 Cs*, http://arxiv.org/abs/1604.07316, Accessed on Apr. 08, 2020.

[10] L. Chi and Y. Mu, Learning end-to-end autonomous steering model from spatial and temporal visual cues, *Proc. of the Workshop on Visual Analysis in Smart and Connected Communities*, Mountain View, CA, USA, pp.9-16, DOI: 10.1145/3132734.3132737, 2017.

[11] C. J. Kim, M. J. Lee, K. H. Hwang et al., End-to-end deep learning-based autonomous driving control for high-speed environment, *J. Supercomput.*, vol.78, pp.1961-1982, 2022.

[12] S. Aditya and S. Mitchell, *Autonomous Driving Using End-to-End Deep Learning: An Airsim Tutorial*, https://github.com/microsoft/AutonomousDrivingCookbook/tree/master/AirSimE2EDeepLearning, Accessed on Apr. 08, 2020.

[13] S. Makridakis, A. Andersen, R. Cabone et al., The accuracy of extrapolation (time series) methods: Results of a forecasting competition, *Journal of Forecasting*, vol.1, pp.111-153, 1982.

[14] A. Burgmeier, C. Lovett, R. Madaan and S. Shah, *AirSim Settings*, https://github.com/Microsoft/AirSim/blob/master/docs/settings.md, Accessed on Apr. 08, 2020.