The Journal of Supercomputing (2022) 78:1961–1982
https://doi.org/10.1007/s11227-021-03929-8

End-to-end deep learning-based autonomous driving control for high-speed environment

Cheol-jin Kim[1] · Myung-jae Lee[1] · Kyu-hong Hwang[1] · Young-guk Ha[1]

Accepted: 2 June 2021 / Published online: 22 June 2021

Abstract
With the recent emergence of artificial intelligence (AI) technology, autonomous vehicle industry has rapidly adopted this technology to investigate self-driving systems based on AI technology. Although autonomous driving is frequently used in high-speed environments, most studies are conducted on low-speed driving on complex urban roads. Currently, most commercialized self-driving cars in SAE autonomous driving level 2 provide practical performance on high-speed roads using various sensors. However, these systems have to process huge sensor data and apply complex control algorithms. Recently, studies have been conducted on the use of image-based end-to-end deep learning to control autonomous driving systems that can be configured at a low cost without expensive sensors and complex processes. In this study, we proposed an autonomous driving control system using a novel end-to-end deep learning model for high-speed environments, and also compared the performance of the proposed system with NVIDIA end-to-end driving system.

Keywords Autonomous driving · End-to-end learning · CNN · LSTM

✉ Young-guk Ha
ygha@konkuk.ac.kr

Cheol-jin Kim
cjfwls1070@naver.com

Myung-jae Lee
dualespresso@naver.com

Kyu-hong Hwang
gfvxgd2k@konkuk.ac.kr

[1] Konkuk University, Seoul, South Korea

1 Introduction

Deep neural networks have led to the development of various industries using computer vision that provides much useful information from the visual data. However, the development of computer vision has been gradual due to the shortcomings such as much information to process, until the development of the deep neural network, particularly the convolutional neural network (CNN). The development of the CNN has made it possible for artificial intelligence-based computer vision to surpass humans in image recognition.

CNN operates using three-dimensional convolution vectors that extract features from images because image data are generally represented in a three-dimensional form with horizontal and vertical edges, and color types such as red, green, and blue. In particular, the convolution operations of CNN can be implemented in parallel graphics processing units (GPUs), which can significantly accelerate training and inference. Using the CNN calculation method and GPUs, computer vision technologies such as object recognition have developed significantly, and various industries using this technology have developed together.

Among different types of industries, the autonomous driving systems that use the advanced AI-based computer vision technology have been explicitly studied. For a long time, many car companies have been working to develop a fully autonomous car, and Google has been experimenting and developing self-driving vehicles for commercialization. The studies on NVIDIA platform for autonomous driving have led to the development of DrivePX that uses the deep learning technology itself. These studies are mainly concerned with the low-speed (less than 60 km/h) driving scenarios in the city. However, in reality, the use of self driving is more common on express vehicle-only roads than in the city where complex intersections, person mobility, and pedestrians exist.

Currently, the SAE Level 2 self-driving system, commercialized for use on express roads, controls autonomous driving using various sensors such as Radar, LiDAR, and cameras. While the use of various sensors can ensure the autonomous driving control performance, the system is very expensive because complex sensor data must be handled, control algorithms must be applied, and the car, including the controller, must be configured. To address these shortcomings, one of the concepts currently being studied is end-to-end learning. It is a way of constructing a network that derives output values according to input values at on-time, eliminating complex intermediate processes. When applied to autonomous driving systems, the end-to-end deep learning system eliminates all complex sensor data processing or control logic and infers driving control values from an input image as the result of the end-to-end network. Because its data processing is based on images without the use of other expensive sensors or processing technologies, it can be configured at a lower cost than that required for currently commercialized self-driving systems. Consequently, in this study, we propose a self-driving control technology for a high-speed driving environment based on end-to-end deep learning.

When controlling self-driving cars in a high-speed environment, it is important that they do not face a situation of understeer or oversteer when they meet a curve section. However, the recently studied end-to-end deep learning-based self-driving control systems, such as NVIDIA, do not consider speed but are mainly focused on steering wheel control only. Controlling the steering wheel without considering speed in a high-speed environment can cause serious problems such as understeer or oversteer. To solve these problems, in this study, we propose a method to control the steering wheel of a self-driving vehicle while considering its speed in a stable manner even in a high-speed driving environment.

Because the conventional autonomous driving system is equipped with many sensors, it is most important to accurately and rapidly process various sensor data of the vehicle. However, to imitate human driving more accurately, it is necessary to utilize image data obtained from the front of the vehicle, instead of relying only on sensors mounted on the vehicle. In addition, it is necessary to analyze how humans use this image data for driving. Humans have the ability to recognize moving objects such as vehicles and pedestrians, and non-moving objects such as traffic lights and roads in front of them while driving. After recognizing these various objects in front, the current vehicle control values such as the driving speed and the steering wheel angle are combined with the situation in front of the vehicle to determine how to steer the wheel and press the accelerator or the brake pedal.

However, there are some critical points to consider, which are not mentioned above. Drivers do not analyze the situation using only a single forward view but continuous views. CNN has the strength to extract spatial features because it calculates using three-dimensional vectors, but ineffective in temporal feature extraction due to its structure. Therefore, it is impossible to develop an autonomous driving system that imitates human driving using CNN only.

Previous studies have learned driving image and driving control value using CNN-based end-to-end models. Through this, it is possible to infer the driving control value from the driving image. However, we thought that the process of driving was difficult to judge by looking at only one fragmentary image. And we proposed a network that combines CNN and LSTM to learn the driving process rather than a fragmentary image. When human drives a vehicle, he does not drive only by looking at the road screen at the moment. Humans are aware of driving information such as current speed and road conditions ahead. And based on this, humans judge how to drive next and drives. As such, it is not considered that an end-to-end model using only CNN can perfectly learn about driving. Therefore, we designed a network that can infer the next driving control value by learning the current driving information and the previous driving information, and this model shows better performance.

In this study, we propose an end-to-end deep learning-based autonomous driving system using both CNN and long short-term memory (LSTM) to extract spatial and temporal features. This study shows a novel end-to-end deep learning model for autonomous driving in high-speed environments.

2 Related works

Similar to the system proposed in this study, various systems have been proposed to infer the steering angle of autonomous vehicles based on CNN. NVIDIA proposed an end-to-end learning system using CNN in 2016 [1]. This system infers the steering wheel angle value by receiving images from the front through three cameras. It is much simpler and more accurate than the conventional methods of detecting lanes or path planning. Their system is trained to minimize the mean squared error of the recorded steering wheel angle and output of the network. The NVIDIA's CNN network has nine layers, approximately 27 million connections, and 250,000 parameters. It effectively infers the steering wheel angles, but do not consider the vehicle speed. Furthermore, it uses CNN only and do not consider the time-series data. Therefore, the angle control may be unstable at high speeds.

In 2017, Chen et al. proposed an end-to-end learning model for the lane keeping of self-driving cars [2]. The experiment was conducted using a CNN. Their system is similar to the NVIDIA's architecture for inferring the steering angle values using the features extracted from the images. It showed high performance, but did not consider the current speed and time-series data like the NVIDIA's system.

A CNN-based steering wheel inferring system is efficient to design an autonomous vehicle system. However, because it infers the steering angle value of the vehicle without considering the vehicle speed, problems such as understeer or oversteer may occur when the vehicle is actually driven. These problems are fatal for an autonomous driving system.

Yang et al. suggested an end-to-end multi-modal multi-task vehicle control model using CNN and LSTM in 2018 [3]. They proposed this CNN-based system that infers the steering wheel angle value from the extracted image features. Unlike the NVIDIA's system, the system infers the vehicle speed value not only using the steering wheel angle, but also LSTM. It is similar to the system proposed in this study because both the steering angles and the speed values are provided as the input for the network, and both CNN and LSTM are used to draw the relevant inferences. However, Yang et al. designed the system using separate neural networks to infer the steering wheel angle and the vehicle speed. Moreover, unlike our work that uses speed as an input to infer the steering angle value, the former system focused on speed control using image features as an intermediate input to infer the speed value.

As shown in Table 1, other studies similar to ours have also used end-to-end learning, but the steering wheel angle value was inferred based on a single CNN

Table 1 Comparison of related works

	NVIDIA	Chen et al.	Yang et al.	Our work
Input	Image	Image	Image/speed	Image/speed
Output	Steering angle	Steering angle	Steering angle/speed	Steering angle/speed
Network type	CNN	CNN	CNN/LSTM	CNN/LSTM
Considering speed to infer angle	No	No	No	Yes

while considering the speed value or the time-series data. Alternatively, the speed and the steering wheel angle were used together in the network, but these studies did not consider speed when inferring the steering wheel angle, focusing only on drawing the speed inference.

In this study, we suggest a system for safe autonomous driving control at a high speed using speed values that were not considered to infer the steering wheel angle in the conventional studies.

Li, You, and Javier Ibanez-Guzman suggested ideas on how to use LiDAR technology in autonomous vehicle systems [4]. They proposed a method for Object Detection, Object Tracking, and Object Intention Prediction by learning the range view of LiDAR through DNN, such as RageNet. And also, Kim, Jinwoo, et al. have proposed a technology that recognizes and tracks moving objects using various sensors in an autonomous driving system [5]. The experiment was conducted on a limited embedded hardware platform, but showed high performance in real time at least 15 fps. In previous papers, various systems that are helpful for autonomous driving systems have been proposed. In this paper, we propose a method of implementing the various systems through end-to-end learning. Grigorescu Sorin, et al. proposed on End-to-End Learning Control that takes various sensor data as input and outputs vehicle control values [6]. It is similar to our proposal in that the intermediate processes, such as Perception, Localization, Path Plannign, Behavior Arbitration, and Motion Control can be solved through End-to-End Learning. However, it differs from this paper in that it is a survey paper without experiments and that it uses all of various sensors, not just driving images. In this paper, there is a strength on implementing an autonomous driving system at low cost by using only driving images without using other sensor data.

3 Background

3.1 Understeer and oversteer

Figure 1 shows the understeer and oversteer scenarios. Understeer is a phenomenon in which the rotation angle of the vehicle becomes smaller than the angle of the

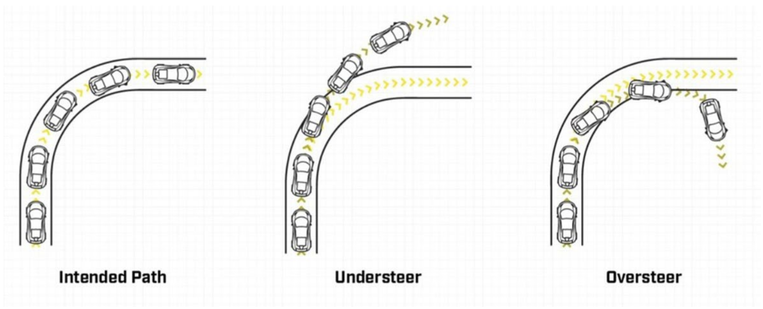


Fig. 1 Visualization of understeer and oversteer

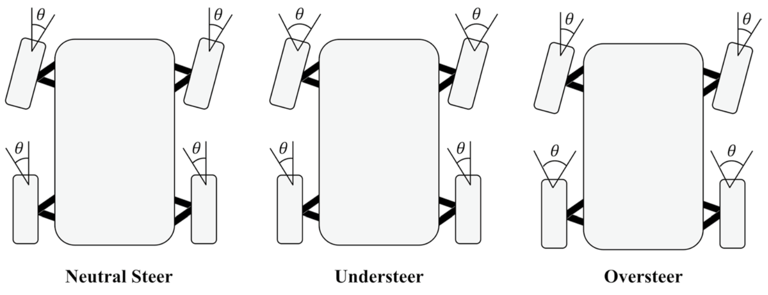


Fig. 2 Explanation of understeer and oversteer based on slip angle

steering wheel when it turns around a corner. Furthermore, from the perspective of the tire grip, understeer occurs when the front tire starts to lose grip before the rear tire during cornering. The front tire already reaches the limit of its grip and cannot increase the lateral acceleration further; therefore, it turns at a larger angle than the intended angle. Conversely, oversteer occurs when the rear tire has less grip than the front tire during cornering.

As shown in Fig. 2, understeer and oversteer can be explained using the concept of the slip angle from the perspective of vehicle dynamics. The slip angle is defined as the angle between the direction in which the wheel is actually going and the direction in which it is currently pointing. The trend direction of the tire is reversed in the direction of wheel rotation because of the friction on the grip surface of the tire. Thus, understeer occurs when the front slip angle of the vehicle is greater than its rear slip angle.

Preventing understeer and oversteer is simple: simply control the appropriate speed of the vehicle before entering a corner. In this study, we propose a method to infer the steering wheel value of an autonomous vehicle by considering the driving speed.

3.2 Convolutional neural network (CNN)

The concept of CNN was first introduced by LeCun in 1989 [7]. At the time, although the use of this concept in handwriting recognition showed significant results, there was insufficient evidence to generalize it. After 9 years, in 1998, LeCun proposed a network called LeNet [8] that became the first CNN. In his study, he designed LeNet, the basis for CNNs, according to the following three ideas: local receptive fields, shared weights, and sub-sampling. Later, in 2014, AlexNet [9] that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was proposed. AlexNet introduced a variety of ideas that are currently being studied in the field of CNNs. Unlike LeNet-5, AlexNet uses a three-dimensional vector considering an RGB image, GPUs to perform several calculations, and ReLU as an activation function. Furthermore, overfitting was solved by applying Max pooling and Dropout on AlexNet. ZFNet [10] that won the ILSVRC in 2013 is a method to find the optimal CNN structure through visualization using deconvolution.

Google's GoogleNet [11] was proposed in 2014 and is a popularly known deep neural network. GoogleNet consists of 152 layers through the inception structure and won the ILSVRC 2014. It was the beginning of the deep CNN.

3.3 Long short-term memory (LSTM)

A recurrent neural network (RNN) is a neural network that can effectively infer time-series data. It uses a short pre-inference value as the input to draw the inference. Because it uses the pre-inference value as the input, it has an advantage in inferring time-series data. However, as the network deepens and learns several processes over time, it faces a vanishing gradient problem: the previously inferred value gradually loses its meaning.

LSTM was proposed by Hochreiter et al. in 1997, and it solved the above-mentioned problem [12]. It is a type of RNN that can perform long learning sessions. The core idea is the cell state; it can store the necessary data for the long-term using an input gate, a forget gate, and an output gate. This cell state solved the vanishing gradient problem. Thereafter, a gated recurrent unit (GRU) was proposed in 2014 [13], which has a simpler structure than LSTM.

As mentioned earlier, LSTM was created to address the vanishing gradient problem of RNN. The cell state plays the role of a conveyor belt, and thus, the gradient is relatively well propagated even after the state has elapsed for a considerably long period of time.

The detailed formula is as follows: $\odot$ denotes the Hadamard product calculation that means multiplication by element.

$$f_t = \sigma * (W_{wh_f} * x_t + W_{hh_f} * h_{t-1} + b_{h_f}) \tag{1}$$

$$i_t = \sigma * (W_{xh\_i} * x_t + W_{hh\_i} * h_{t-1} + b_{h\_i}) \tag{2}$$

$$o_t = \sigma * (W_{xh\_o} * x_t + W_{hh\_o} * h_{t-1} + b_{h\_o}) \tag{3}$$

$$g_t = \tanh(W_{xh\_g} * x_t + W_{hh\_g} * h_{t-1} + b_{h\_g}) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

The forget gate f_t is the gate to forget past information. It is activated by a sigmoid function receiving h_{t-1} and x_t as inputs. Because the output range of the sigmoid is between 0 and 1, if the output of the forget gate is 0, the previous status information is forgotten, and if it is 1, the previous status is completely remembered.

The input gate $i_t \odot g_t$ is the gate to remember the current information. It executes the sigmoid function with h_{t-1} and t_t as inputs, and then executes the tanh function with the same inputs. Thereafter, it performs the Hadamard product operation and prints the output value of the input gate.

Through these operations, LSTM solves the vanishing gradient problem.

3.4 Long-term recurrent convolutional networks (LRCNs)

CNN has certain advantages with respect to extracting features using a two-dimensional vector, but is not efficient in obtaining both temporal and spatial information. To solve this problem, many studies combined CNN that extracts spatial information and RNN that extracts temporal information. The popularly known LSTM is a fully connected LSTM structure in which both the input and the output are one-dimensional vectors. To infer from the image on LSTM, which is a three-dimensional vector including channel, it is necessary to transform the model structure.

Xingjian et al. proposed the convolutional LSTM in 2015 [14]. It uses a three-dimensional vector for the input and output of each cell, and its internal operation was replaced by performing the convolution as a simple matrix multiplication. Consequently, the spatial and temporal features can be extracted simultaneously from the cell itself. It is also efficient, but in this study, we used the long-term recurrent convolutional networks (LRCNs) proposed in 2015 [15]. This model first extracts features from an image, converts them into one-dimensional vectors, and inserts them into the input of LSTM.

In this study, the spatial and temporal features occurring during driving were extracted using the LTRCN model to determine the driving plan.

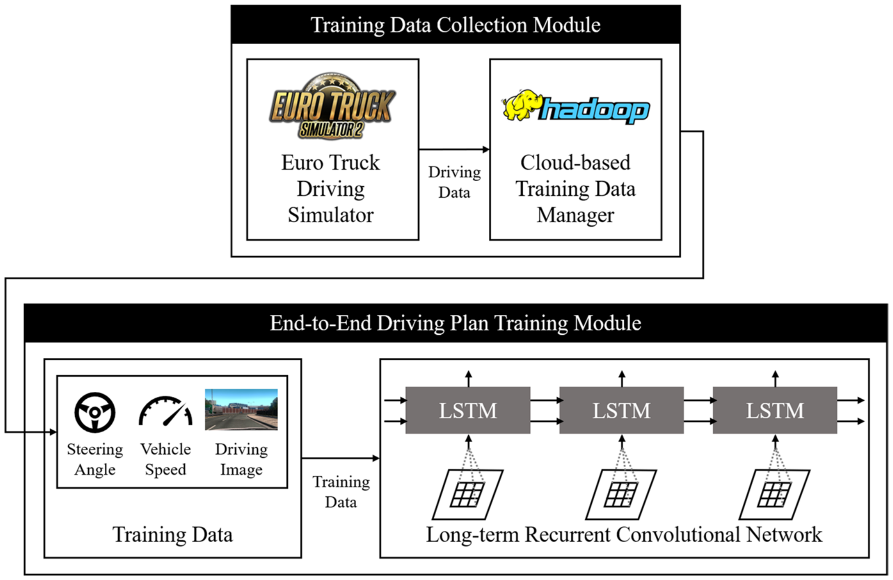

Training Data Collection Module
EURO TRUCK SIMULATOR 2
Euro Truck Driving Simulator
Driving Data
hadoop
Cloud-based Training Data Manager
End-to-End Driving Plan Training Module
Steering Angle
Vehicle Speed
Driving Image
Training Data
Training Data
LSTM
LSTM
LSTM
Long-term Recurrent Convolutional Network

Fig. 3 Overall system architecture

4 System design

Figure 3 shows the overall architecture of the proposed system. The training data collection module collects the training data from the Euro truck simulator and saves the data on the cloud-based training data manager. The end-to-end driving plan training module trains the weights of the network via the CNN and LSTM using a training set from the cloud server. The system infers a driving plan, including the steering wheel angle and the vehicle speed.

4.1 Training data collection module

The training data collection module collects the driving data using the Euro truck simulator for use in the training dataset. We designed the driving data collection system, as shown in Fig. 4. The Euro truck simulator-based data collection module

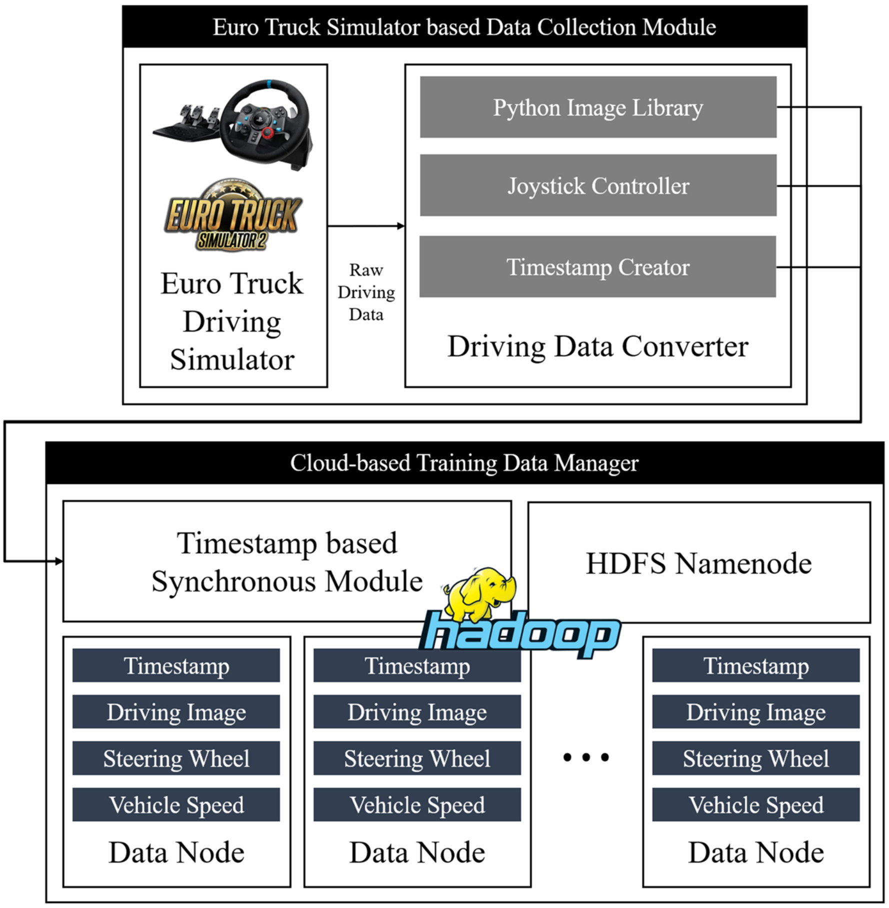


Fig. 4 Driving data collection system

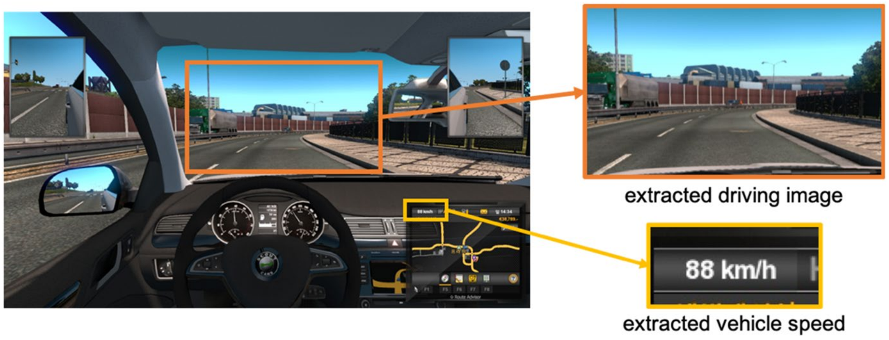


Fig. 5 Extracting driving image and vehicle speed

collects the raw data from the euro truck simulator. The raw data contain time-seriesed image with timestamp and steering wheel value between – 1.0 and 1.0. The driving data converter receives raw driving data from the euro truck simulator and divides them into di erent types of data such as the steering wheel, speed, timestamp, and the driving image. The cloud-based training data manager receives the converted data and synchronizes it with a timestamp- based synchronous module, and saves the data to DataNode.

In the driving data converter, with the Python Image Library, we cropped a driving image for use in the training. As shown in Fig. 5, the driving image and the vehicle speed have been extracted from a raw driving image. The system analyzed the vehicle speed using a simple CNN.

4.2 End-to-end driving plan training module

As shown in Fig. 6, the overall architecture of the driving plan training module consists of two parts: training dataset and driving plan training using the LRCN. The training dataset is loaded from the driving data cloud server mentioned above. As shown in the right panel of Fig. 6, the training module trains the weights of the

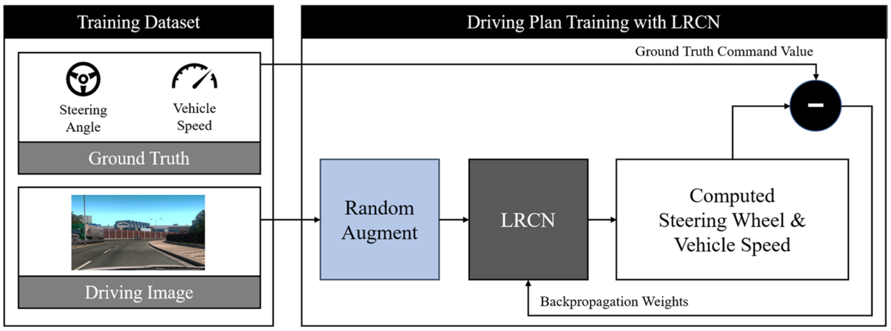


Fig. 6 Driving data collection system

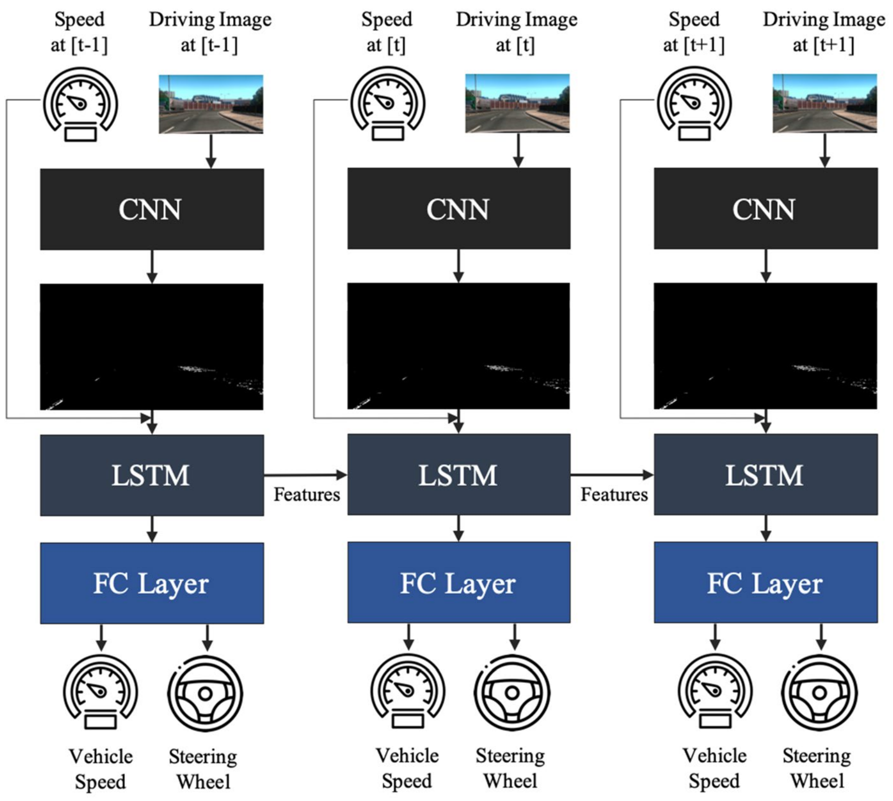


Fig. 7 Overall system flow of training driving plan

network to minimize the mean square error between the ground-truth driving plan and the computed one using the LRCN

4.2.1 End-to-end driving plan training module

The overall training model structure is shown in Fig. 7. The CNN extracted the features of the images. After the concatenation of the speed data, the extracted features became the input values of LSTM. The steering wheel value and the vehicle speed were inferred through the LSTM cell and the fully connected layer. The feature maps extracted from the CNN were reused to infer the driving plan of the next time series. By reusing these feature maps, the system inferred a driving plan considering changes over time.

Figure 8 shows the structure of the end-to-end deep learning training model. The input driving image and speed required for learning were normalized before they were entered as the inputs. First, the driving image underwent a simple feature extraction, with a total of five convolution layers. The features of the image extracted through five

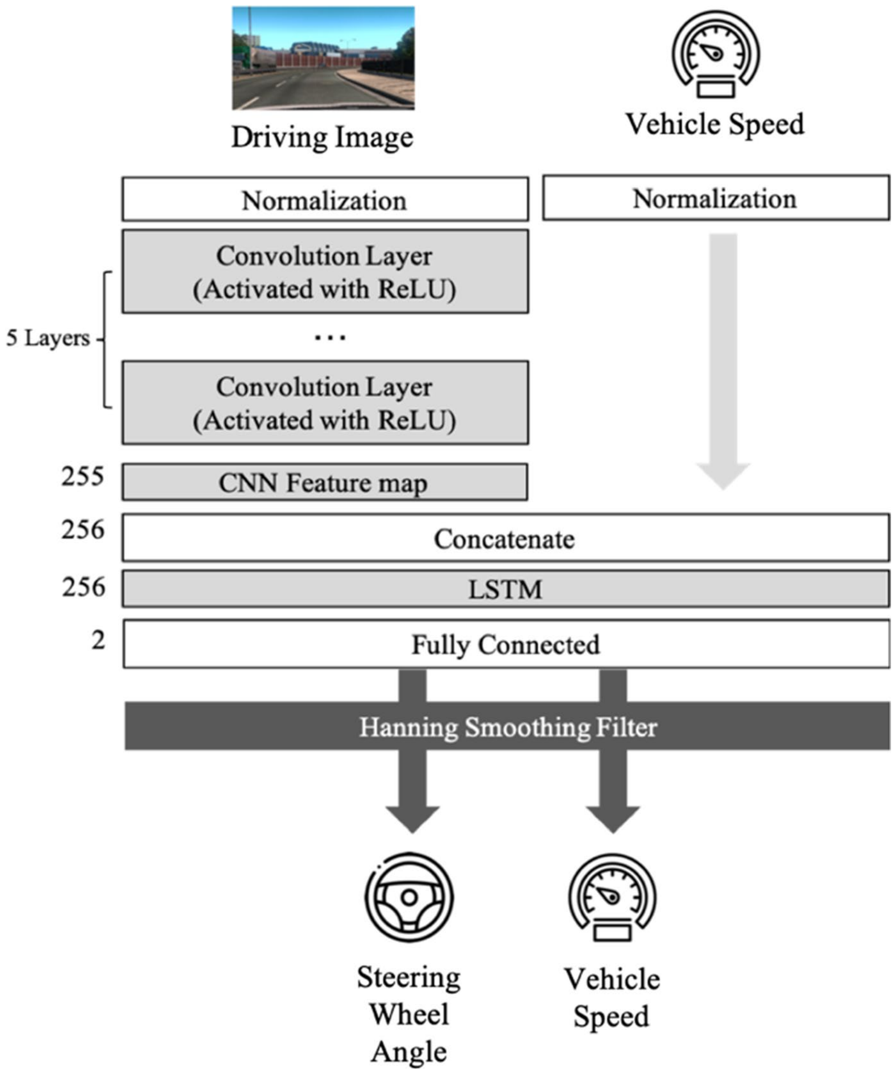


Fig. 8 Training model architecture

convolution layers were concatenated with the normalized speed value. The concatenated vector became an input to the LSTM and was passed through the LSTM cell to produce a one-dimensional vector output with 256 sizes. Finally, the steering wheel angle and speed were inferred from a fully connected layer. The inferred values were converted using the Hanning smoothing algorithm for the smoothing of the continuous angle and speed.

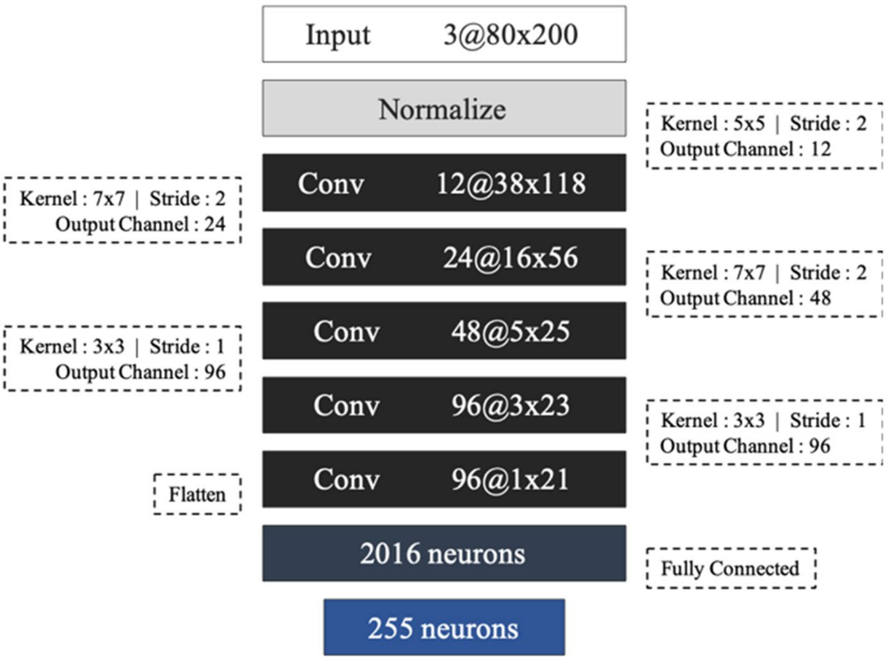


Fig. 9 Feature extraction using CNN

4.2.2 Extracting features from images using CN

Figure 9 shows the network structure of the CNN part, which extracts features from the images. Before performing convolution, input image is resized to an 80×200 vector, and then normalized. The normalized vector was passed through a total of five different layers for integration. The padding of the convolution layer used in the experiment was 1, and the activation function was ReLU. A feature vector of 255 sizes was created with seven layers, including the flatten and fully connected layers. The extracted feature vector was concatenated with the normalized speed vector, and then used as an input to the LSTM.

4.2.3 Angle and speed prediction using LSTM

As shown in Fig. 10, LSTM infers the final output vector by receiving a feature extracted from the mentioned CNN and a speed vector as the input. For experiments, 40-frame sequences were used to draw the inference. The output vector was used as the input to the next stage considering the changes over time.

4.2.4 Loss function for training

We used mean square error (MSE) as a loss function for training driving plan. The formula of the loss function applied to this system is as follows.

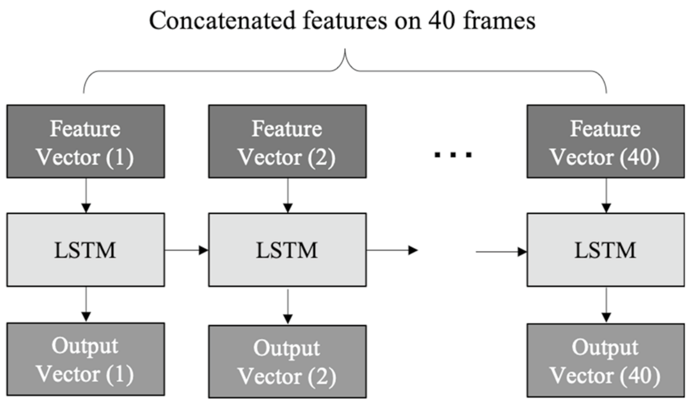


Fig. 10 Inferring angle and speed using LSTM

$$\mathrm{loss}_{\mathrm{angle}} = \frac{\sum_{i=1}^{n}(Y_{\mathrm{angle}\_i} - y_{\mathrm{angle}\_i})^2}{n} \tag{7}$$

$$\mathrm{loss}_{\mathrm{speed}} = \frac{\sum_{i=1}^{n}(Y_{\mathrm{speed}\_i} - y_{\mathrm{speed}\_i})^2}{n} \tag{8}$$

$$\mathrm{loss} = \frac{\mathrm{loss}_{\mathrm{angle}} + \mathrm{loss}_{\mathrm{speed}}}{2} \tag{9}$$

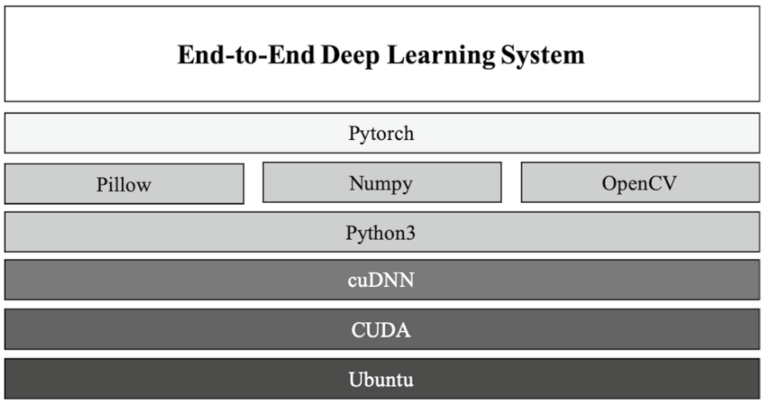


Fig. 11 Software architecture of the proposed system

4.2.5 Software architecture

The software architecture of the proposed system is shown in Fig. 11. To use the GPUs, the CUDA and cuDNN were installed on the base, and the Python Library and Python were installed. We used the Pytorch framework to develop the end-to-end deep learning system.

5 Experiments

5.1 Dataset

We used the Euro truck simulator with the Logitech G29 steering wheel and pedals to collect the driving dataset. We collected approximately 400 GB of converted training data to infer the steering wheel angle and the vehicle speed.

5.2 Training

We constructed a deep learning server for the end-to-end deep learning training. And we used two GPUs with RTX2080ti, and AMD Ryzen 2950x CPU with 16 cores and 32 threads.

To use LSTM for the training, the sequence length of the images had to be fixed. The experimental results revealed that the best sequence length was 40 frames. The sequence of a frame was just over 1 s when we considered 30 fps, yielding the best learning result when the sequence of this length was entered.

5.3 Evaluation

5.3.1 Performance evaluation with ground truth

As shown in Fig. 12 the steering wheel predictions are almost the same as the ground truth. Because of the specificity of LSTM to use the pre-inference data to infer the next step, some errors occurred in the early stage because of the lack of

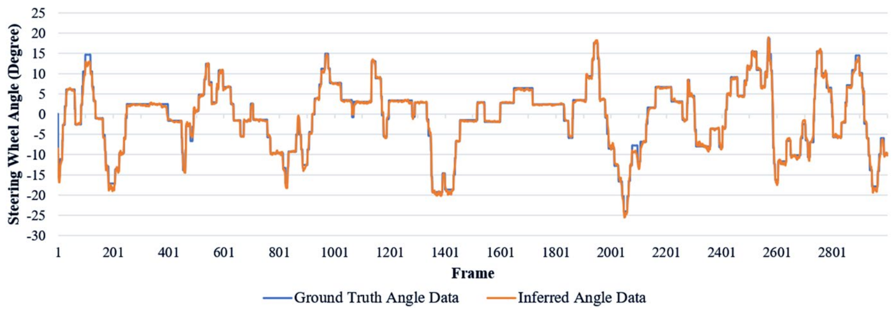


Fig. 12 Accuracy of inferring a steering angle for 3000 frames

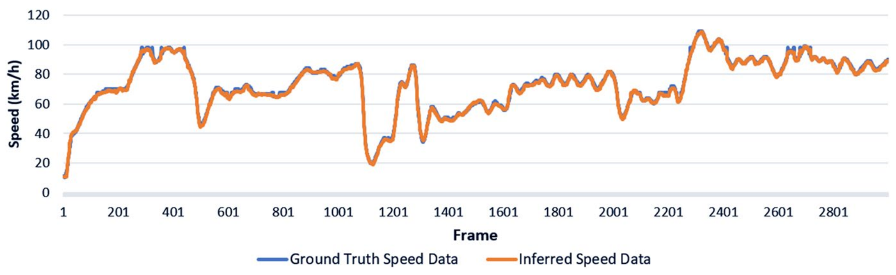


Fig. 13 Accuracy of inferring a speed for 3000 frames

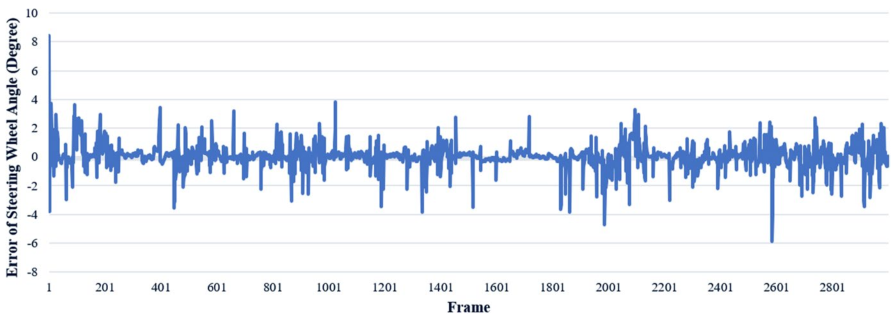


Fig. 14 Error in inferring the steering angle for 3000 frames

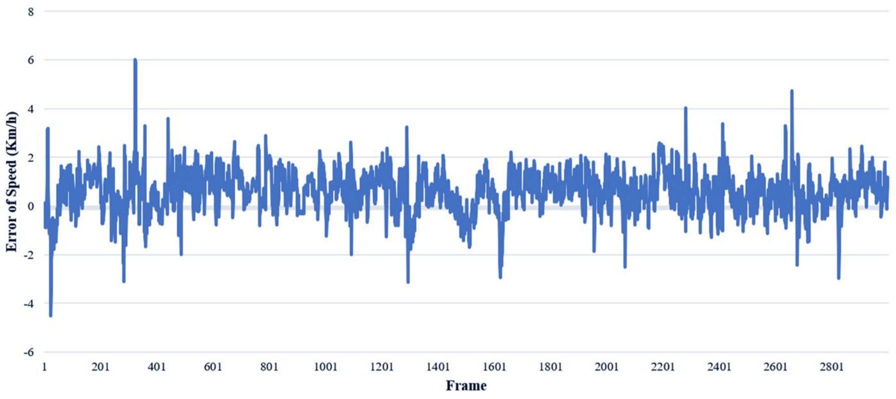


Fig. 15 Error of inferring a speed for 3000 frames

initial data. Similar to the steering angle value prediction, the vehicle speed prediction was also observed, as shown in Fig. 13.

The errors of the steering angle and speed for 3000 frames are shown in Figs. 14 and 15, respectively.

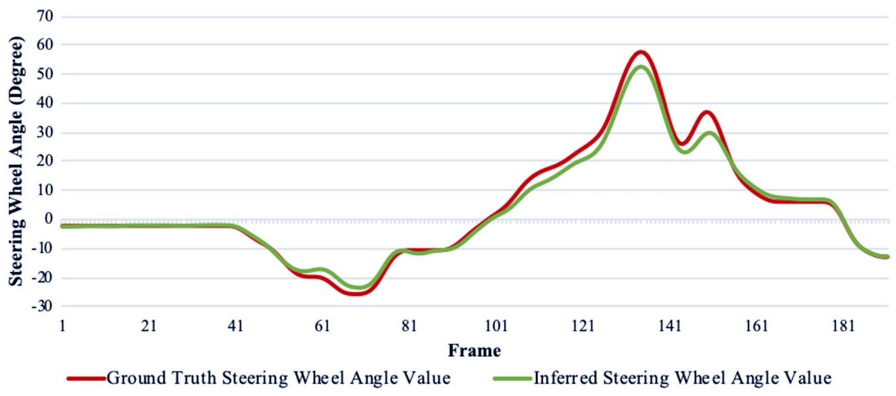


Fig. 16 Accuracy of inferring the steering wheel angle on S-curved road

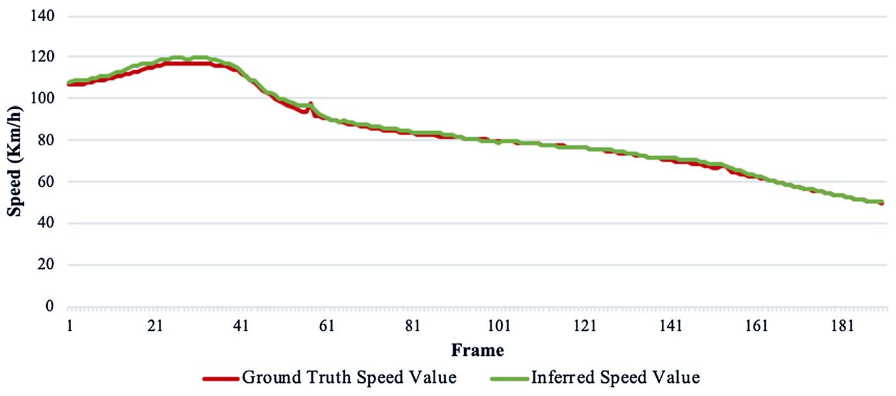


Fig. 17 Accuracy of inferring the speed on S-curved road

Considering that the steering angle ranged from – 450° to 450° and the speed ranged from 0 to 120 km/h, this system yielded very accurate prediction results with an average steering angle error of 0.028 and an average speed error of 0.65.

The graphs in Figs. 16 and 17 show the accuracy of inferring the steering wheel angle and the speed on an S-curved road, respectively. Even though the vehicle was driven on a curved road, our network inferred the appropriate angle and speed for safe driving.

5.3.2 Performance evaluation by comparison with NVIDIA's work

Under general driving conditions that do not involve a curve with a large angle, the experimental results presented in this study and those obtained using the NVIDIA's network were similar. As shown in Fig. 18, the performance of both the networks is similar.

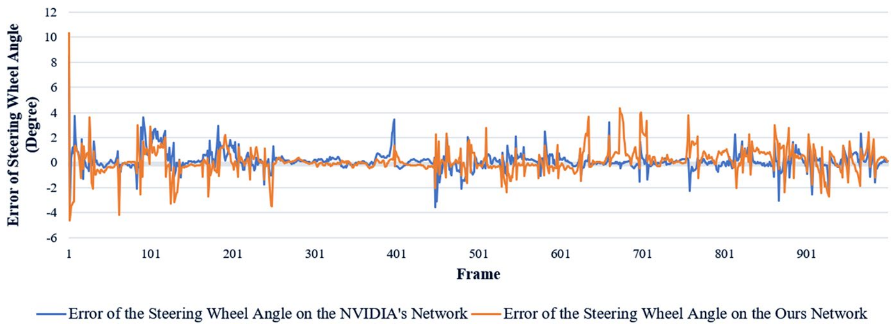


Fig. 18 Error of steering angle prediction in two networks

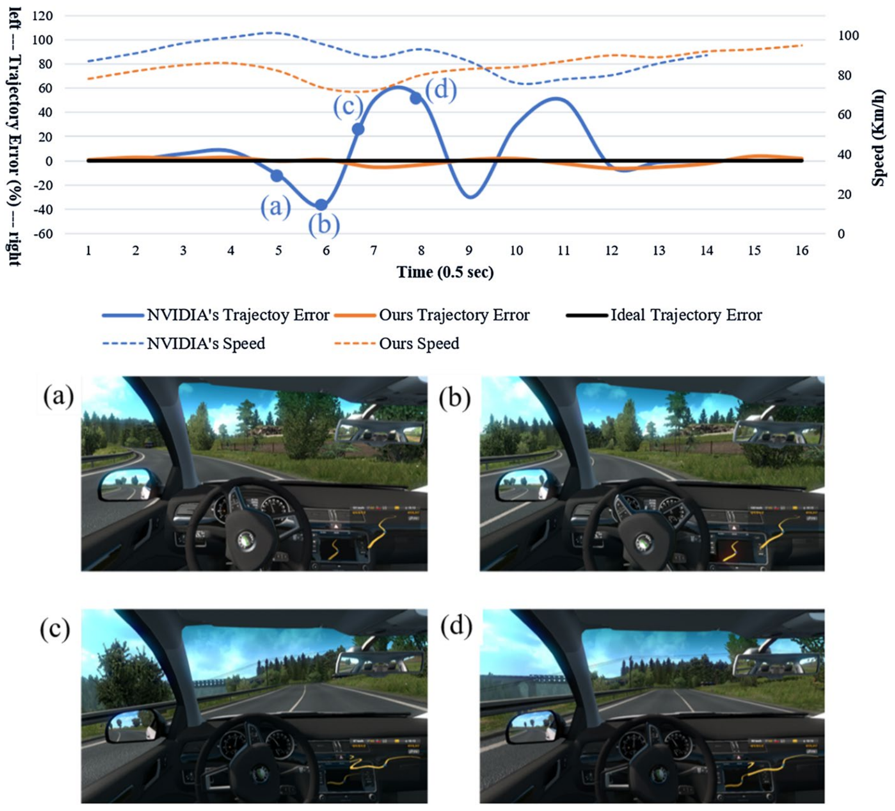


Fig. 19 Self-driving simulation results on a left curved road

However, some differences occurred on the curved road while driving at a high speed. The NVIDIA's network predicted a steering angle close to the generated understeer because, unlike our network, the NVIDIAs network was trained without considering the vehicle speed. Figure 19 shows the results of the trajectory error and speed caused as a result of the control in the proposed network and the NVIDIA's network when a high-speed vehicle was driven on a left-curved road. The NVIDIA's network inferred the steering wheel angle without considering and controlling the speed, whereas the proposed system inferred the appropriate speed and the speed-considered steering wheel angle. Consequently, the trajectory data inferred from the proposed network showed stable values and those from the NVIDIA's network showed gradual understeer and oversteer phenomena. The time-series driving images are shown in Fig. 19. Furthermore, Figs. 20 and 21 show the other results for a right curved road and a right-left curved (S-curved) road, respectively.

These results of these experiments showed that it was safer to infer the steering wheel angle considering the time-series image and the current speed of the vehicle under high-speed conditions than not.

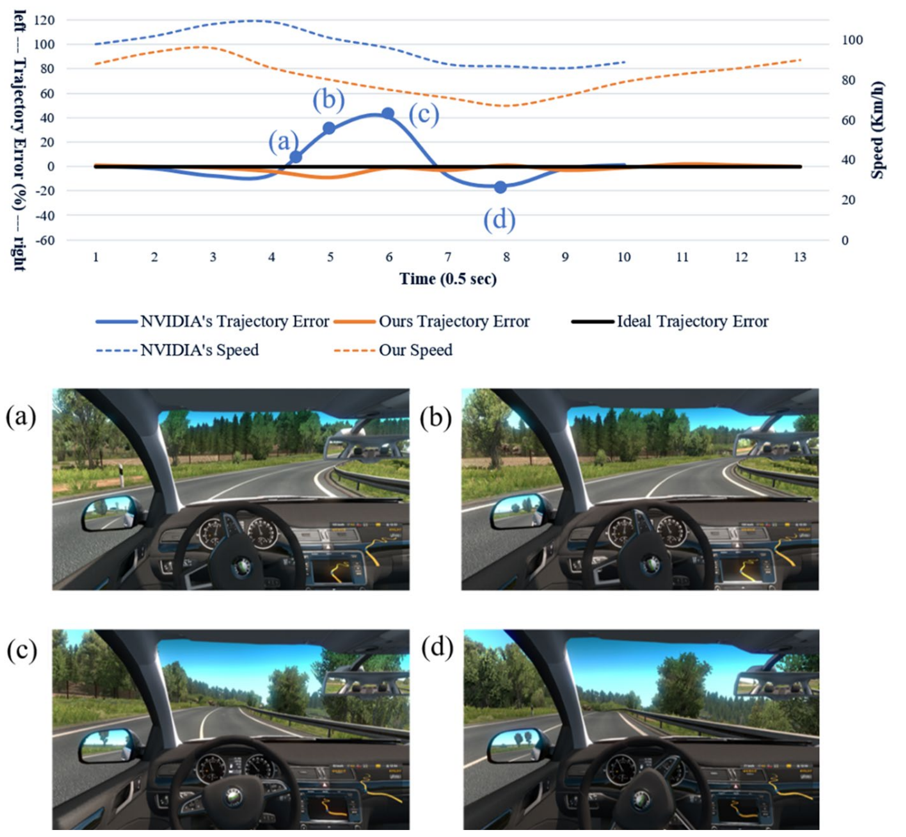
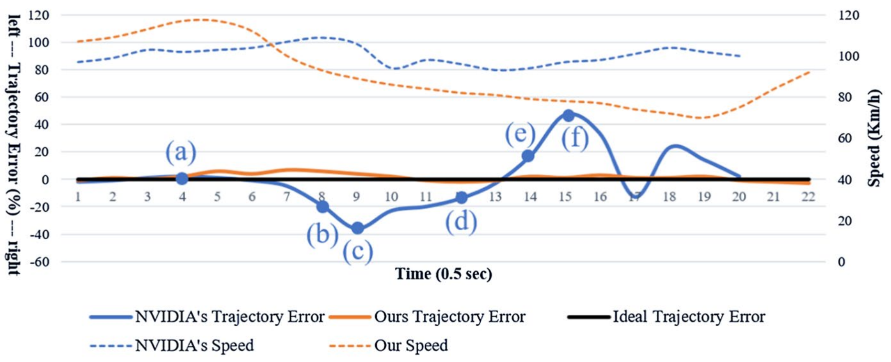


Fig. 20 Self-driving simulation results on a right curved road



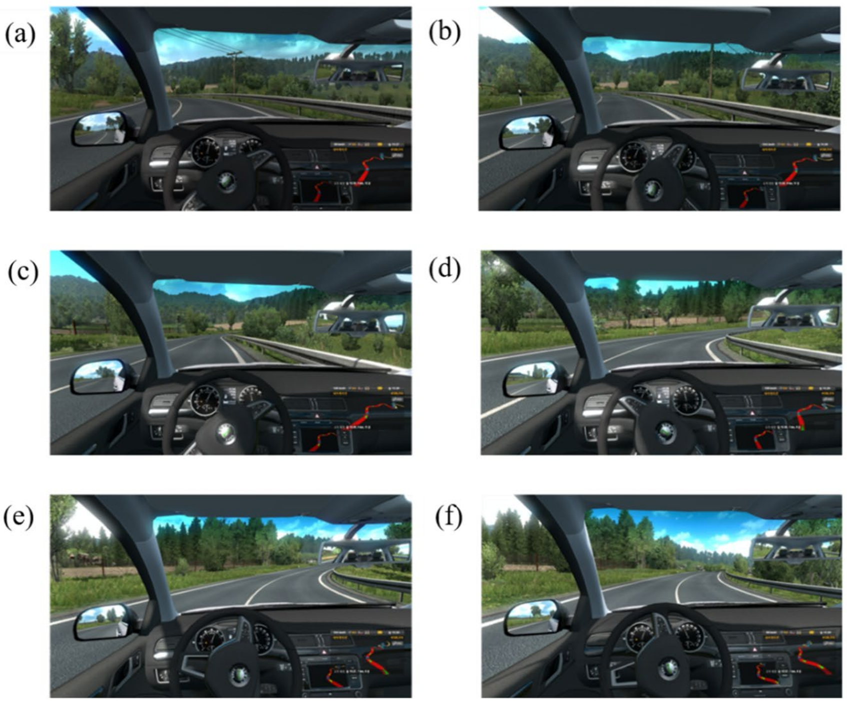


Fig. 21 Self-driving simulation results on an S-curved road

5.3.3 Implementation of results using smoothing algorithm

The angle values inferred from the end-to-end driving networks showed high accuracy, but splashes occurred when compared to the ground truth because of inferring the angle of each successive frame. It could be improved by applying the smoothing algorithm using the Hanning window value, as shown in (10). As

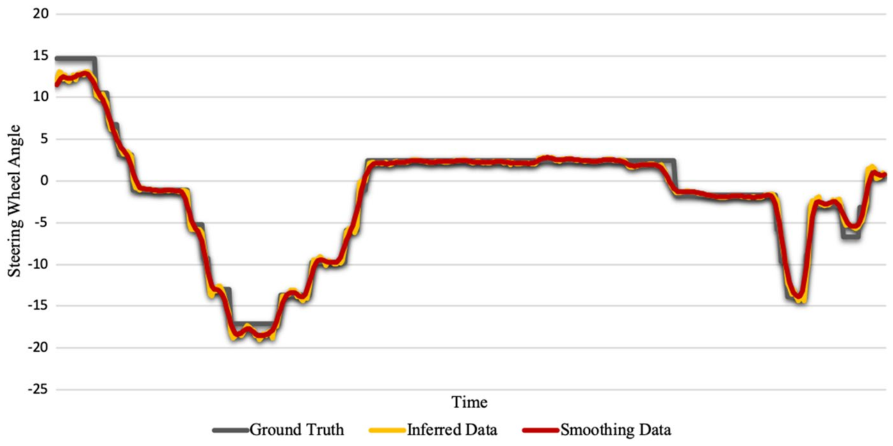


Fig. 22 Accuracy of inferring steering angle using smoothing algorithm

shown in Fig. 22, the graphs with the smoothing algorithm have a smooth curve and can be observed to be closer to the ground truth.

$$w(n) = 0.5 - 0.5\cos(\frac{2\pi n}{M-1}), 0 \le n \le M-1 \tag{10}$$

6 Conclusion

In this study, we proposed an end-to-end autonomous driving system by incorporating CNN and LSTM. This combination of CNN and LSTM allowed the extraction of both the temporal and the spatial features. In addition, the steering wheel angle values and the speed values of the vehicle could be inferred by considering the changes over time. It also demonstrated the effect of preventing dangerous situations such as understeer in a high-speed environment by inferring the driving speed and the steering wheel angle together, while considering the current speed when inferring the steering wheel angle.

Acknowledgements This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2016-0-00465) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

References

1. Bojarski M et al (2016) End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316
2. Chen Z, Huang X (2017) End-to-end learning for lane keeping of self-driving cars. In: 2017 IEEE Intelligent Vehicles Symposium (IV). IEEE

3. Yang Z et al (2018)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). IEEE
4. Li Y, Ibanez-Guzman (2020) Lidar for autonomous driving: the principles, challenges, and trends for automotive lidar and perception systems. arXiv preprint arXiv:2004.08467
5. Kim J et al (2019) Multi-sensor-based detection and tracking of moving objects for relative position estimation in autonomous driving conditions. J Supercomput 76:8225–8247
6. Grigorescu S et al (2020) A survey of deep learning techniques for autonomous driving. J Field Robot 37(3):362–386
7. LeCun Y et al (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
8. LeCun Y et al (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
9. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems
10. Zeiler MD, Rob F (2014) Visualizing and understanding convolutional networks. In: European Conference on Computer Vision. Springer, Cham
11. Szegedy C et al. (2015)Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition
12. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
13. Chung J et al (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555
14. Shi X et al (2015) Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems
15. Donahue J et al (2015) Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition