sustainability

MDPI

Article

Advanced Camera Image Cropping Approach for CNN-Based End-to-End Controls on Sustainable Computing

Yunsick Sung [1], Yong Jin [1], Jeonghoon Kwak [1], Sang-Geol Lee [2] and Kyungeun Cho [1,*]

[1] Department of Multimedia Enginnering, Dongguk University-Seoul, 30 Pildong-ro, 1-gil, Jung-gu, Seoul 04620, Korea; sung@dongguk.edu (Y.S.); j.yong@dongguk.edu (Y.J.); jeonghoon@dongguk.edu (J.K.)

[2] Department of Electrical and Computer Engineering, Pusan National University, 2 Busandaehak-ro, 63 Beon-gil, Geumjeong-gu, Busan 46241, Korea; leesg@pusan.ac.kr

* Correspondence: cke@dongguk.edu; Tel.: +82-2-2260-3834

Received: 1 February 2018; Accepted: 9 March 2018; Published: 15 March 2018

Abstract: Recent research on deep learning has been applied to a diversity of fields. In particular, numerous studies have been conducted on self-driving vehicles using end-to-end approaches based on images captured by a single camera. End-to-end controls learn the output vectors of output devices directly from the input vectors of available input devices. In other words, an end-to-end approach learns not by analyzing the meaning of input vectors, but by extracting optimal output vectors based on input vectors. Generally, when end-to-end control is applied to self-driving vehicles, the steering wheel and pedals are controlled autonomously by learning from the images captured by a camera. However, high-resolution images captured from a car cannot be directly used as inputs to Convolutional Neural Networks (CNNs) owing to memory limitations; the image size needs to be efficiently reduced. Therefore, it is necessary to extract features from captured images automatically and to generate input images by merging the parts of the images that contain the extracted features. This paper proposes a learning method for end-to-end control that generates input images for CNNs by extracting road parts from input images, identifying the edges of the extracted road parts, and merging the parts of the images that contain the detected edges. In addition, a CNN model for end-to-end control is introduced. Experiments involving the Open Racing Car Simulator (TORCS), a sustainable computing environment for cars, confirmed the effectiveness of the proposed method for self-driving by comparing the accumulated difference in the angle of the steering wheel in the images generated by it with those of resized images containing the entire captured area and cropped images containing only a part of the captured area. The results showed that the proposed method reduced the accumulated difference by 0.839% and 0.850% compared to those yielded by the resized images and cropped images, respectively.

Keywords: self-driving; convolution neural network; end-to-end control

1. Introduction

Research on utilizing sensors, such as radar and light detection and ranging (LiDAR), attached to vehicles has been actively under way to accurately recognize surrounding environments for self-driving vehicles [1]. The recognized surrounding environment information is used to control the vehicles. For example, vehicles are controlled based on fuzzy logic [2] or motor primitives [3].

As sensors for self-driving vehicles are generally expensive, various methods for deriving environmental information using relatively low-cost cameras have been attempted. A traffic sign area is derived [4] or a traffic sign is recognized [5] to recognize the road on which a vehicle is driving. Images captured by cameras attached to both sides of a vehicle have been utilized for analyzing the

environment so that the vehicle can drive in keeping with its lane [6]. Additionally, a method has been proposed to derive the position of the lane on which a vehicle is driving using the captured images of the area in front of the vehicle [7]. Using images captured by a camera mounted on a vehicle, obstacles are recognized [8,9] or the intentions of pedestrians on the road are predicted [10]. Furthermore, research has been conducted on sharing the environment information measured by one vehicle with other vehicles [11].

Recently, studies on end-to-end control-based self-driving have been actively conducted to control vehicles using images captured by one or multiple cameras attached to them as input [12–14]. In general, the entire process of enabling a vehicle to drive autonomously consists of the following steps: recognizing obstacles, deciding on a driving direction based on recognized obstacles, and controlling the vehicle based on the decided driving direction. However, end-to-end control has an advantage in that it controls the vehicle by learning vehicle control signals based on input images without the need for the traditional process of recognition, decision, and control. Various deep learning techniques have been applied to learning for end-to-end control. Vehicles operate autonomously by learning from images of the road entered into convolutional neural networks (CNNs) [12–14]. Studies have been conducted on autonomous driving using deep neural networks (DNNs) [15] and on predicting the driving distance of vehicles using recurrent convolutional neural networks (RCNNs) [16]. However, high-resolution images cannot be entered into CNNs directly due to memory limitations and need to be adjusted for end-to-end control. In other words, the high-resolution images should be cropped or resized such that they retain sufficient information for self-driving vehicles. Existing learning methods involve manual extraction of the area in front of vehicles from captured images [14] or use an optimal area by learning based on multiple learning areas directly set by users [17]. However, there is a need for a method that can automatically derive the area without users selecting the learning areas, to enable autonomous driving in various environments.

This paper proposes a method to generate input images for CNNs. Multiple parts of images featuring both sides of a lane are extracted and merged, by considering the perspective of the car, as input images. This reduces the learning time needed for self-driving cars through end-to-end controls. A CNN model is also proposed for the system to learn by using the generated input images. Based on the cropped images and the proposed CNN model, the process of the system learning how to control the car is described. Furthermore, an approach to infer the angles of the steering wheel to control the car using cropped images is introduced.

The structure of the remainder of this paper is as follows: Section 2 introduces related work in the area and Section 3 describes the image generation method for CNNs in detail. Section 4 discusses the experimental results to validate the proposed method and Section 5 contains the conclusions of this paper.

2. Related Works

2.1. Research on Awareness of Cars

Research has been conducted on the recognition of traffic signs on the road [4,5]. Based on images captured by a self-driving car, a traffic sign is recognized using a CNN [5]. The hinge loss stochastic gradient descent approach has been proposed to train the CNN. The target zone for detecting the traffic sign is filtered by a color-based thresholding algorithm. The traffic sign shape is then identified using the distance-to-borders (DtBs) approach. Zainal et al. [4] propose recognizing traffic signs using an artificial neural network (ANN) based on two feature descriptors, the histogram-oriented gradient (HOG), and speeded-up robust features (SURF). The velocity of the car is controlled based on the detected traffic signs.

Studies have been conducted to estimate the location of a lane for self-driving cars [6,7]. The lane location on both sides of the car was estimated by DeepLanes [6] using cameras. The location of a given lane was divided into 316 locations and learned using the captured images. The lane with the highest

probability among the 316 lane locations was determined to be the real one. Jiman et al. [7] investigate ego lanes using a camera facing the front of a car. A deep learning model was trained using the data to identify ego lanes in the collected images, and the location of the lane was estimated based on the deep learning model.

Furthermore, Sebastian et al. [8] detect obstacles in front of a car using cameras mounted on it. A three-dimensional (3D) stixel was created using the semantic and geometric features of the data collected by the cameras. Additionally, stereo estimation was used to recognize obstacles in front of a car using a single camera [9], where images taken at the front of a car were used to train a CNN.

Researchers have also attempted to identify the surroundings of self-driving cars [1,10]. Moving objects are recognized using multiple radars, LIDAR, and vision sensors mounted on the car. Depending on the distance between an object and the car, and the type of the object, shape data are provided. Friederike et al. [10] identifies the behaviors of pedestrians using a support vector machine (SVM).

A self-driving car requires expensive systems to detect its surroundings. An approach is needed to determine the direction of motion of the car without lanes. It is also necessary to plan driving routes to avoid obstacles in the environment.

2.2. Research on Driving Method of Cars

An approach has been proposed to estimate the motion of self-driving cars [18]. The direction of motion of a car can be estimated using cameras mounted on it. Studies have also examined self-driving using the deep Q-network [19] where a Q-value is updated through simulations. A car autonomously runs using the Q-value, and learning is needed to update it.

Unghui et al. [20] plan routes for self-driving cars, where the car autonomously operates by considering obstacles. The car checks for obstacles along its path. In case an object is detected along the route, a new self-driving route is created to avoid collision in light of the estimated collision point.

Furthermore, studies have investigated self-driving cars based on driving commands and captured images [12–14]. While being driven, the car learns a model for self-driving based on the images and the driving commands. Spikes are extracted from the images collected, and the car learns a spiking neural network model based on these images as input data and the driving commands as output data. The driving commands are then estimated through the spiking neural network by extracting the spiked images from the input images.

2.3. End-to-End Controls of Cars

Visual odometry (VO) is a method of calculating the position changes of a vehicle using images and sensors attached to the vehicle [16]. To measure the position changes of a vehicle, VO calculates them through steps such as feature extraction, feature matching, motion estimation, and local optimization. The end-to-end control method is used to reduce the number of steps involved in VO. Using the end-to-end method, position changes can be calculated by setting the input of the RCNN as camera images and the output as the pose of the vehicle.

Pen et al. [15] apply the end-to-end control method to autonomous driving on a mountain road using low-cost on-board sensors. A DNN was trained using captured images and vehicle control signals to autonomously control vehicles on the mountain road. It demonstrated that autonomous driving is possible even with low-cost on-board sensors by applying the end-to-end control method.

Owing to environmental constraints in a real environment, end-to-end control cannot train the model fully. To overcome this problem, a virtual simulation environment has been used to train the model them fully [14]. An environment similar to the real environment is set up for a virtual simulation, and captured images and control signals of the driving records of virtual vehicles are collected. A CNN is trained based on the collected images and control signals, and then used to control a vehicle in the real environment.

However, owing to memory limitations, captured images cannot be used as inputs to a CNN. Therefore, an approach is required that can crop input images. To address this issue, a study was conducted to extract part of the images captured by cameras attached to the front of the vehicle and use them for training [13]. To reduce the cost of collecting images during driving, images captured from the right, left, and front cameras attached to the front side of a vehicle are used. The images captured by the right and left cameras are transformed into the same shape as those captured by the front camera using a random shift and rotation. The images transformed from the right and left images and the images created by cropping the front area of the vehicle from the front camera images are collected. A CNN is trained using the collected images and steering angles. The vehicle is then autonomously driven by inferring steering angles from the images created by cropping the front area of the vehicle.

Two frameworks that determine a featuring area have been introduced to reduce the training cost of end-to-end control [17]. Each training result is analyzed after generating and learning various learning images. The first framework trains the CNN with the entire captured image. At the time of execution, the sky area, mountain area, and road area are manually derived from the captured images. Steering angles are inferred by inputting the captured images into the CNN that learned them as input images, by which control results are compared. The second framework divides the entire captured image into the sky area, mountain area, and road area, and trains the CNN with each area. At the time of execution, the results of controlling a vehicle based on the entire captured image are analyzed.

During autonomous driving using end-to-end control, some of the captured images are manually derived for use owing to memory issues. A method is required for automatically deriving these images. In particular, it is necessary to derive them by identifying the parts of images that affect the control of steering angles.

3. Image Cropping Approach for Self-Driving Cars

This section describes the proposed method for cropping images captured by a camera mounted on a car to enhance learning performance for self-driving based on the angles of the steering wheel and the cropped images. The method of estimating the angles of the steering wheel using the CNN model is also explained.

3.1. System Overview

The proposed end-to-end control method controls a steering wheel after learning partially merged images obtained from a camera and the angles of the steering wheel. The proposed method comprises of a data collection phase, a learning phase, and an execution phase.

The data collection phase involves collecting images along the driving direction of a car and extracting parts of the images, merging the parts of the images, and recording the merged images with the angles of the steering wheel as shown in Figure 1. Specifically, the RGB image captured at time t is image i_t^L, $\begin{bmatrix} i^L_{t,1,1} & \cdots & i^L_{t,WIDTH,1} \\ \vdots & \ddots & \vdots \\ i^L_{t,1,HEIGHT} & \cdots & i^L_{t,WIDTH,HEIGHT} \end{bmatrix}$, and the steering wheel angle of the car at t is angle a_t^L. The pixel $i^L_{t,x,y}$ is expressed using RGB colors. The image i_t^L and angle a_t^L generated while driving the car are transferred to a shared memory. The image i_t^L in shared memory is divided and merged into two zones, including the lane considered for self-driving. The merged image ${i'}_t^L$ and angle a_t^L are stored in a database.

In the learning phase, the car learns a CNN as an end-to-end control based on the merged image ${i'}_t^L$ and angle a_t^L stored in the database, as shown in Figure 2. The steering wheel angle a_t^E is inferred using the CNN by comparing with the angle a_t^L, adjusting back-propagation weights, and obtaining an error.

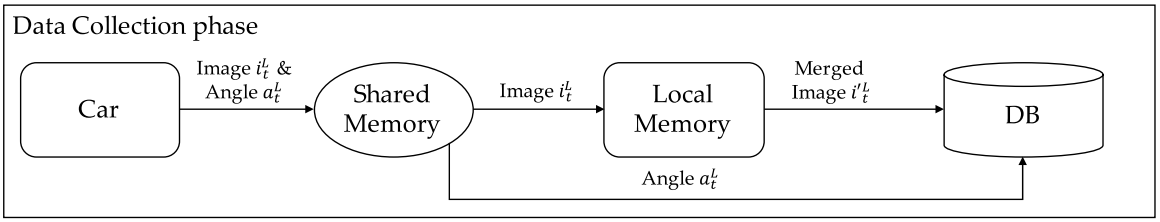


Figure 1. The data collection phase collects cropped images and steering wheel angles generated while a user drives the car.

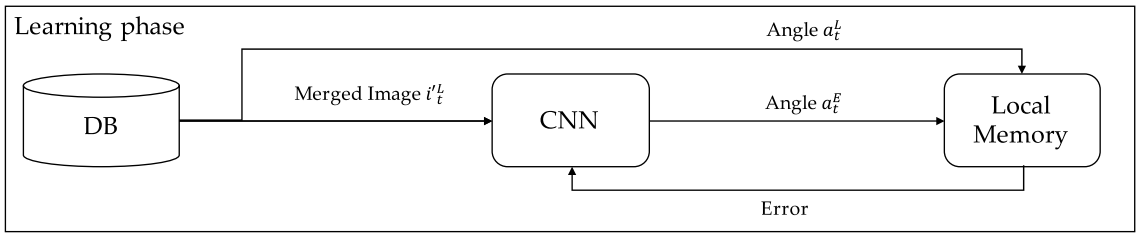


Figure 2. The learning phase learns the CNN using the partially merged images and steering wheel angles collected during the data collection phase.

In the execution phase, the images captured and merged are entered into the CNN and the corresponding steering wheel angle is inferred, as shown in Figure 3. Specifically, image i_t^E is stored in the shared memory and merged as the image i'^E_t. The merged image i'^E_t is entered into the CNN, and the angle a_t^E is inferred and stored in the shared memory. It is then transferred to the car, where the steering wheel angle is controlled based on it.

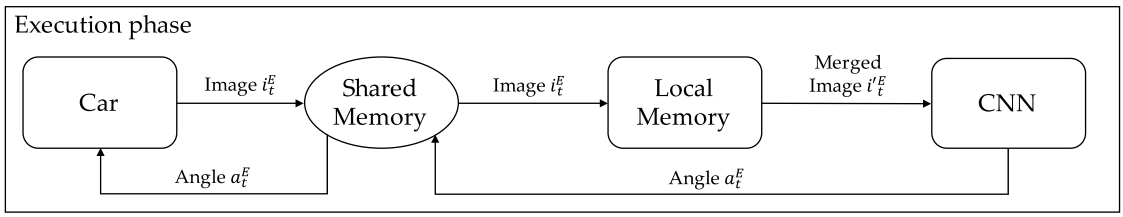


Figure 3. The execution phase crops the images and enters them into the CNN. The car operates using the steering wheel angle as output data of the CNN.

3.2. Extracting and Merging Approaches

Generally, autonomous cars consider lanes for driving along a given route. Given that the lanes constitute key information for self-driving, the areas including the lanes from images captured by a camera mounted on the car should be extracted and utilized. The proposed method extracts lanes as shown in Figure 4.

First, set a road cell that contains only the road parts in a captured image with a rectangle as shown in Figure 4a. The color contained in the road cell is recognized and processed as the road henceforth.

The following steps are taken to calculate the number of pixels corresponding to the road for each pixel position in the captured image. The captured images are converted to gray images to simultaneously process the RGB colors of the captured images. Each pixel of the converted gray image i_t^L is represented by $g_{t,x,y}^G$. The maximum and minimum values of the gray pixels in the road cell are derived from

all the captured images. Therefore, $i''_{x,y}$ is $\sum_{t=1}\begin{cases} 1\ if\ MIN(g^G_{t,x,y}) \leq g^G_{t,x,y}\ and\ g^G_{t,x,y} \leq MAX(g^G_{t,x,y}) \\ 0\ if\ MIN(g^G_{t,x,y}) > g^G_{t,x,y}\ and\ g^G_{t,x,y} > MAX(g^G_{t,x,y}) \end{cases}$.

The intermediate image i''^G_t is $\begin{bmatrix} i'_{1,1} & \cdots & i'_{WIDTH,1} \\ \vdots & i'_{x,y} & \vdots \\ i'_{1,HEIGHT} & \cdots & i'_{WIDTH,HEIGHT} \end{bmatrix}$ as shown in Figure 4b.

Gray images are generated using the intermediate images as shown in Figure 4c. $i'_{x,y}$ is normalized and quantitated by Equation (1). The generated gray image i''^G_t $\begin{bmatrix} i''_{1,1} & \cdots & i''_{WIDTH,1} \\ \vdots & i''_{x,\,y} & \vdots \\ i''_{1,HEIGHT} & \cdots & i''_{WIDTH,HEIGHT} \end{bmatrix}$.

$$i''_{x,y} = \left\lfloor \frac{\alpha_{MAX} \times ((i'_{x,\,y} - MIN(i'_{x,y}))/(MAX(i'_{x,y}) - MIN(i'_{x,y})) \times \alpha_{MAX} + \alpha_{MIN})/\left(\frac{\alpha_{MAX}+1}{\beta}\right)}{(\beta - 1)} \right\rfloor \quad (1)$$

where α_{MAX} and α_{MIN} are the maximum and minimum numbers of available gray colors and β is the quantization constant.

Edges are derived by applying the Canny edge detection algorithm based on the gray image i'^M [21]. In the proposed method, the area containing the road cell is assumed to be the road area, as shown in Figure 4d.

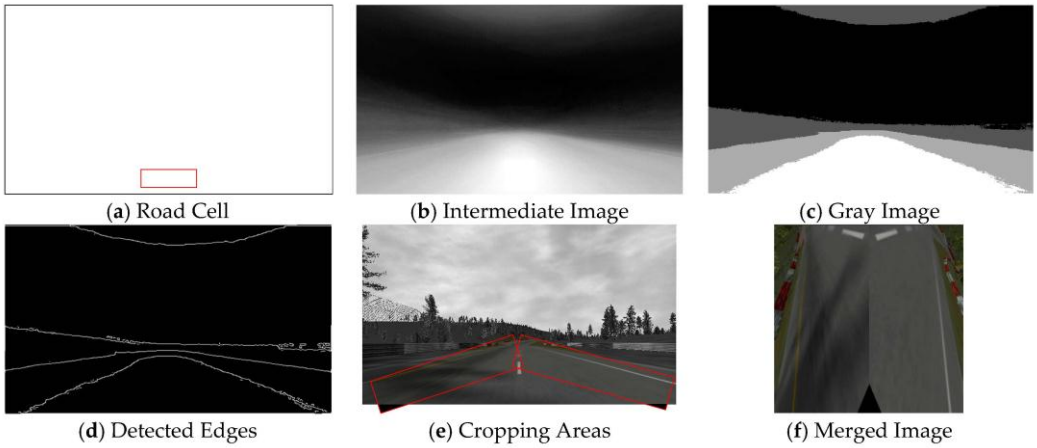


Figure 4. The input images of CNN are generated by extracting lanes and merging selected areas. The road cell is selected and the number of pixels corresponding to the road in the road cell is calculated for each captured image. A gray image based on the number of pixels is created and edges are extracted. Cropping areas are determined by utilizing edges and images are merged. A 200 × 200 RGB image is generated by cropping images obtained from cameras of the car.

Cropping areas are calculated as follows based on the derived edges. In the gray image i'^M, the bottom left point of the road area is defined as edge point e_l, the bottom right point is defined as point e_r, and the middle point is defined as edge point e_m, as shown in Figure 5a. Two rectangles are generated with width γ. The middle of the upper edge of the two rectangles is placed at the edge point e_m, the middle of each lower edge is placed at edge point e_l, and the bottom right point is placed at edge point e_r. The created rectangles are utilized as cropping areas. The cropping areas are resized to the size of the merged image, and then the final merged image is generated as shown in Figure 5b.

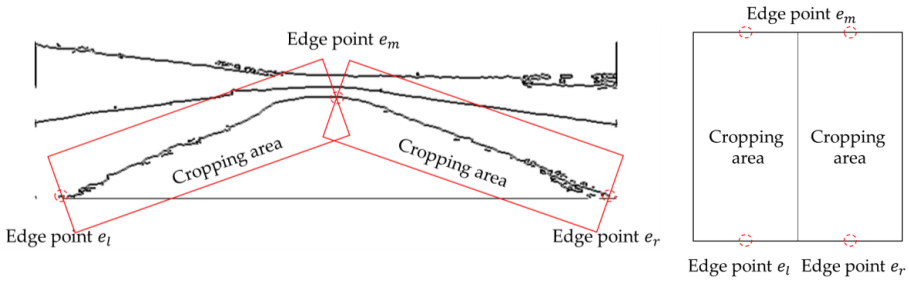


Figure 5. Cropping areas are derived using the boundary. Input images are generated to CNN by merging images based on the two derived cropping areas. (**a**) Derived cropping areas; and (**b**) the final merged image.

3.3. Network Architecture

The CNN structure in the proposed system is shown in Figure 6, and was based on an improvement on AlexNet [22]. However, the angle rather than the classification was inferred. Thus, Euclidean distance was used as the loss function.

The RGB image taken by the camera on the car was converted into a 200 × 200 RGB image. The result of cropping only the lanes considered for self-driving was normalized and used as input data. The input layer of the CNN utilized the 200 × 200 merged image. The CNN comprises five convolution layers, three max pooling layers, and two fully-connected layers. Convolution layers 1, 2, and 3 were estimated using a 5 × 5 kernel, and convolution layers 4 and 5 were estimated using a 3 × 3 kernel. Max pooling layers 1, 2, and 3 were estimated using the 3 × 3 kernel. As the output, the angle a_t^E was returned.

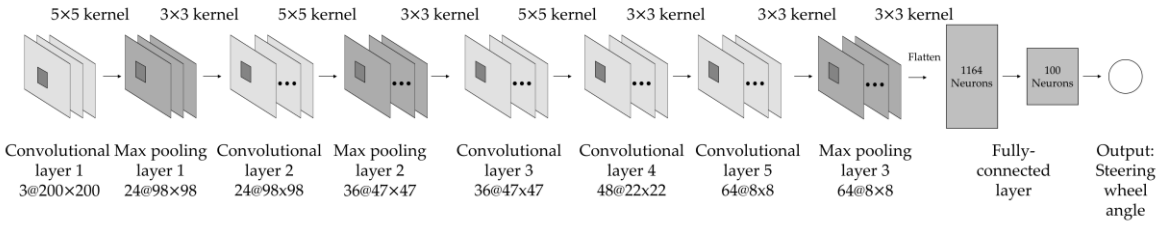


Figure 6. The CNN is configured to enter a 200 × 200 merged image to output the steering wheel angle.

4. Experiments

In experiments, the steering wheel angle using a cropped image from those taken was inferred. The results concerning the steering wheel angle were compared and analyzed using an approach that crops images using a traditional method based on the images captured from the car and that involves resizing of the entire image.

4.1. Experimental Environment

The Open Racing Car Simulator (TORCS) [23] was used to create the driving environment for the car. TORCS is an open-source 3D car racing simulator. It was developed using C++ and provides multiple platforms. Table 1 presents the hardware used for the experiments, which were executed on Ubuntu 16.04. The deep learning library used was Tensorflow, an open-source software [24]. Shared memory was used for interlocking TORCS with Tensorflow. OpenCV (Intel Corporation, Santa Clara, USA) [25] was used for cropping the images on TORCS.

Table 1. Hardware specifications to learn the CNN.

Items	Content
Development Platform	Ubuntu 16.04 (Canonical, London, UK)
Tool	Pycharm 2017.1.4 (JetBrains, Prague, Czech Republic)
CPU	i7-6850 K CPU@3.60 GHz (Intel Corporation, Santa Clara, USA)
GPU	NVIDIA TITAN XP 12 GB × 4 (Nvidia Corporation, Santa Clara, USA)
Memory	16 GB DDR4
Library	OpenCV 3.1.0, Tensorflow, Share memory

The images and steering wheel angles were collected as the car self-drove using a module in TORCS. The car completed five laps along each of six tracks, as shown in Figure 7, and 370 × 640 RGB images and angles were collected at 10 frames per second.

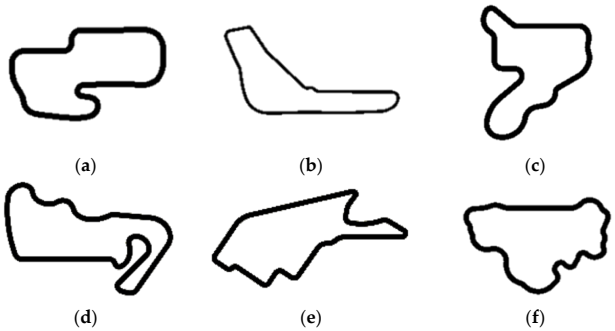


Figure 7. TORCS tracks for data collection. The images and steering wheel angles were collected from a self-driving car using a TORCS module on six tracks. (**a**) Ruunskogen; (**b**) Forza; (**c**) CG track 3; (**d**) Wheel 1; (**e**) Street 1; and (**f**) E-Road.

The 200 × 200 images were created to entering into the CNN from the images taken by TORCS, as shown in Figure 8. Figure 8a shows the original images from the camera of the car. Figure 8b shows the images cropped using the proposed approach and Figure 8c displays the images resized to 200 × 200 pixels [14]. Figure 8d shows the images with the forward field of the car cropped using the traditional approach [13]. Two cropping areas were created with a 232 × 100 image at 71.85 degrees centered on (296, 440) point and a 252 × 100 image at −70.39 degrees centered on the (296, 210) point.

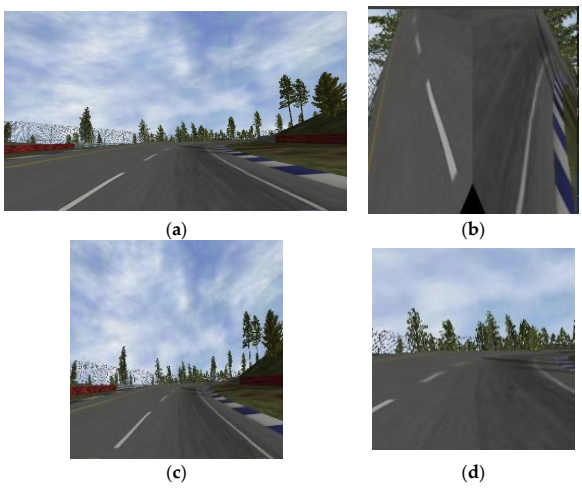


Figure 8. Converted images from those taken on TORCS for input into the CNN. (**a**) The 370 × 640 image was converted into a 200 × 200 image and learned using the proposed CNN configuration. (**a**) The original image; (**b**) the proposed method image; (**c**) the resized image [14]; and (**d**) the cropped image [13].

4.2. Learning Routes

Learning was based on data collected during the self-driving of the car along six tracks for input to the CNN as shown in Figure 7. Learning was carried out using data from the six tracks 30,000 times through a multi-GPU for approximately four hours by changing the inputs. Figure 9 shows the loss according to the number of learning when changing input images. The loss from learning for 10,000 to 28,000 iterations was maintained the proposed approach, the resized image and that involving the cropping of the forward field. The result of the loss converged to 0.01 with 28,000 iterations in three approaches.

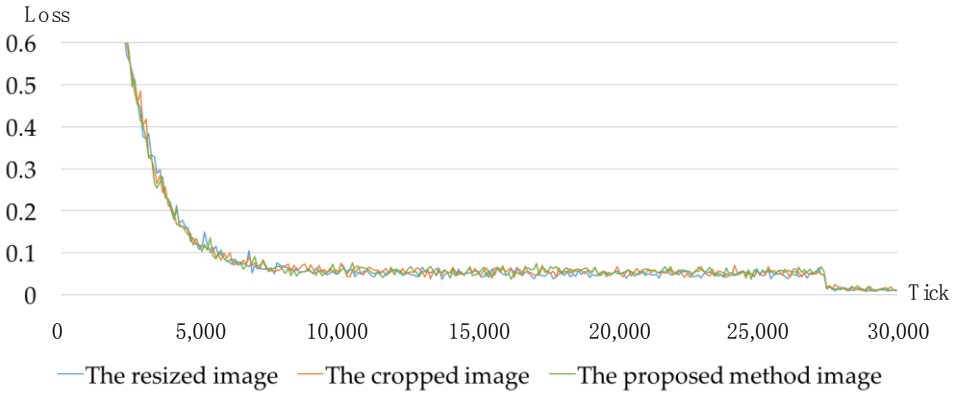


Figure 9. Change in CNN loss when learning CNN configuration on six tracks using three images.

4.3. Experimental Results and Performance Analysis

Figure 10 illustrates the self-driving course of the TORCS car using the steering wheel angle created by the CNN learned through the six tracks. The car ran along the course using the learned CNN by employing the image generated by the proposed method, the resized image, and the cropped image.

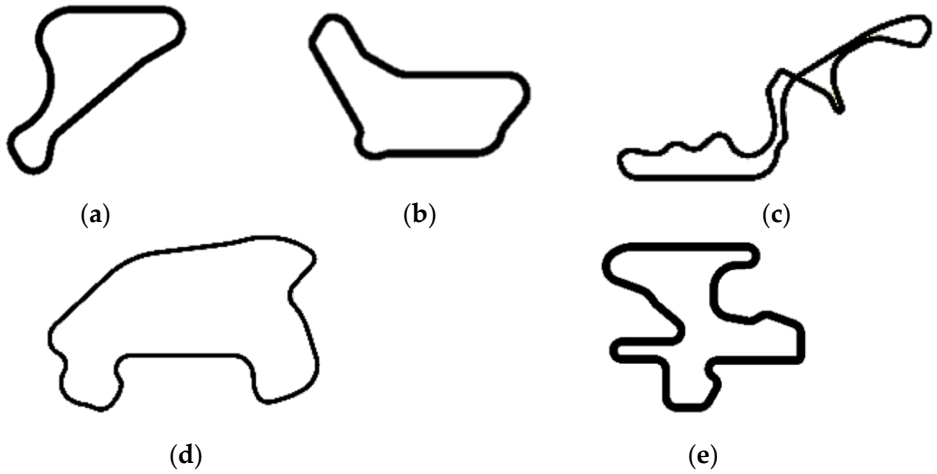
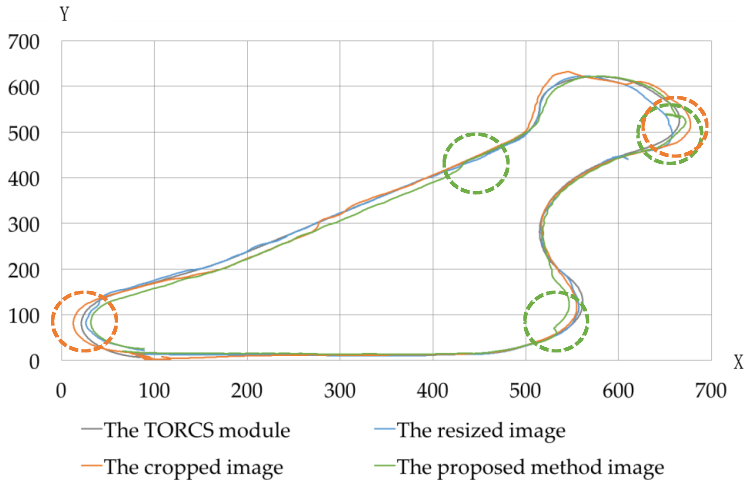


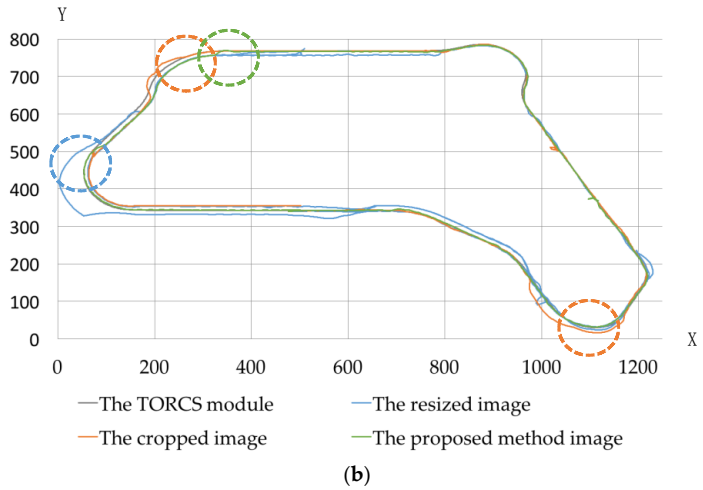
Figure 10. Final course determined to analyze the driving results using the learned CNN. The self-driving test was conducted using the CNN learned with the image obtained by the proposed method, the resized image, and the cropped image. (**a**) CG Speedway number 1; (**b**) CG Track 2; (**c**) Wheel 2; (**d**) E-Track 4; and (**e**) Alpine 2

Figure 11 shows the results for the course in Figure 10. The car using the proposed approach moved differently in the straight course, along the guard rail. CNN learning using the cropped image failed to follow the route where the curve started and collided into the guard rail. The resized

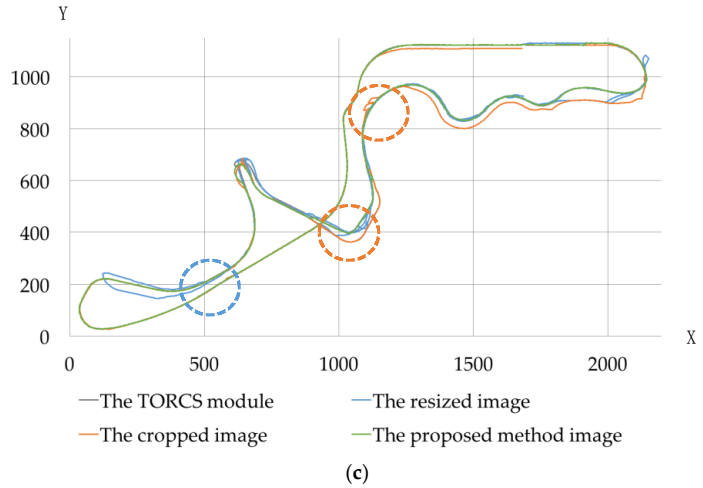
images were smaller than the cropped image but were found to deviate from the curve on the course. In Figure 11c–e, the car no longer ran along the course and collided in place and no longer proceeded. On the contrary, the proposed approach successfully allowed the self-driving car to run along the curve.



(**a**)



(**b**)



(**c**)

Figure 10. *Cont.*

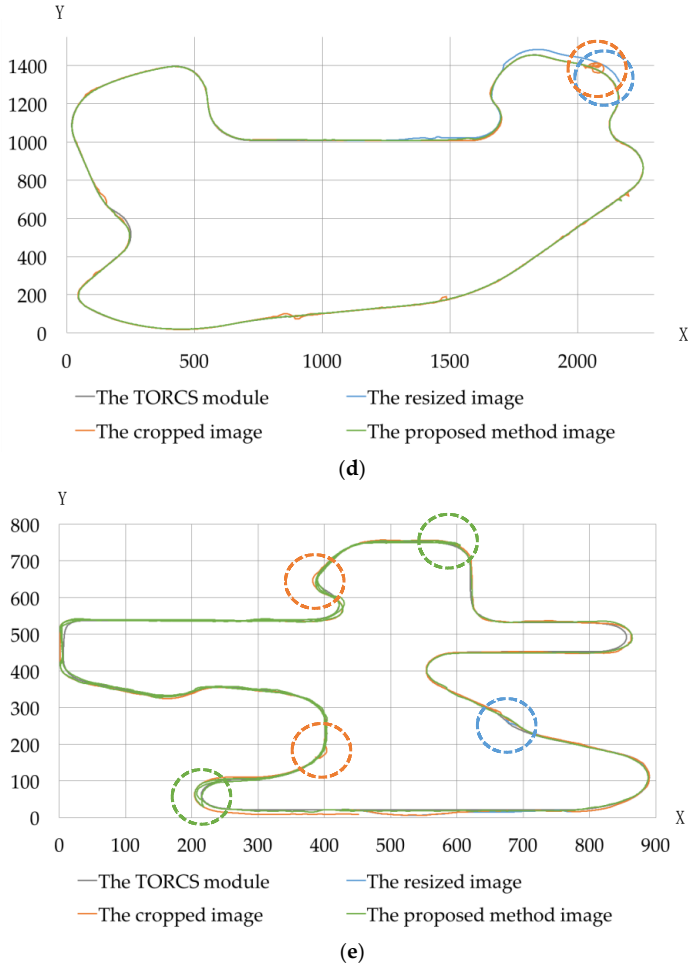


Figure 11. The results of self-driving along the course in Figure 10. The car moved along the course using the CNN learned with the image generated by the proposed method, the resized image, and the results were obtained using a module in TORCS. (**a**) CG Speedway number 1; (**b**) CG Track 2; (**c**) Wheel 2; (**d**) E-Track 4; and (**e**) Alpine 2

Images and steering wheel angles were collected by TORCS for the course in Figure 10. The steering wheel angles of the three methods were compared and the accumulated results are shown in Table 1. The average cumulative error of the proposed approach was 499.936, that of the method using resized image was 595.672, and the error when cropped image was used was 587.785. The proposed method has the smallest cumulative error, except for Table 2.

Table 2. The accumulated difference between the steering wheel angles collected in TORCS and the steering wheel angles generated by three approaches. Three approaches are the proposed method, the resized image, and the cropped image.

Method	CG Speedway Number 1	CG Track 2	Wheel 2	E-Track 4	Alpine 2
The proposed method	791.5418	320.236	593.2503	301.958	492.6927
The resized image [14]	833.7867	448.4071	744.6954	426.2724	525.1988
The cropped image [13]	912.9649	406.769	650.2188	278.4263	690.5489

5. Conclusions

This paper proposed a method to crop input images for self-driving cars using end-to-end control. Areas in the images representing lanes were extracted from the images collected during self-driving

for use as input to the end-to-end control. The images containing the lanes were used as the input to the CNN and the steering wheel angle was set as the output.

The experiment involved the CNN learning the images and steering wheel angles during the self-driving on TORCS. The steering wheel angles were then inferred. It was verified that the learning result improved using the proposed cropping areas. The proposed method was 0.839% better than the approach using the resized images containing the entire area of the original images, and 0.850% better than the approach using cropped images containing only a part of the original images. The learning performance improved by eliminating unnecessary areas of the images using end-to-end control. However, further research is needed to investigate the approach to improve performance on unlearned roads.

Acknowledgments: This research was supported by BK21 Plus project of the National Research Foundation of Korea Grant and by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2013-0-00684) supervised by the IITP (Institute for Information and communications Technology Promotion).

Author Contributions: Yunsick Sung proposed a main idea and wrote the proposed approach. Yong Jin did experiments, Jeonghoon Kwak did additional experiments and is in charging in writing this paper overall, Sang-Geol Lee designed the system based on the proposed approach and adjusted a CNN model, and Kyungeun Cho supervised the main idea and written proposed method.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cho, H.; Seo, Y.W.; Kumar, B.V.; Rajkumar, R.R. A Multi-Sensor Fusion System for Moving Object Detection and Tracking in Urban Driving Environments. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014.
2. Khoobjou, E.; Mazinan, A.H. On Hybrid Intelligence-based Control Approach with its Application to Flexible Robot System. *Hum.-Cent. Comput. Inf. Sci.* **2017**, *7*, 1–18. [CrossRef]
3. Sung, Y.; Kwak, J.; Park, J.H. Graph-Based Motor Primitive Generation Framework: UAV Motor Primitives by Demonstration-based Learning. *Hum.-Cent. Comput. Inf. Sci.* **2015**, *5*, 1–9. [CrossRef]
4. Abedin, M.Z.; Dhar, P.; Deb, K. Traffic Sign Recognition Using Hybrid Features Descriptor and Artificial Neural Network Classifier. In Proceedings of the 19th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 18–20 December 2016.
5. Jin, J.; Fu, K.; Zhang, C. Traffic Sign Recognition with Hinge Loss Trained Convolutional Neural Networks. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 1991–2000. [CrossRef]
6. Gurghian, A.; Koduri, T.; Bailur, S.V.; Carey, K.J.; Murali, V.N. DeepLanes: End-To-End Lane Position Estimation using Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, USA, 26 June–1 July 2016.
7. Jiman, K.; Chanjong, P. End-to-End Ego Lane Estimation based on Sequential Transfer Learning for Self-Driving Cars. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017.
8. Ramos, S.; Gehrig, S.; Pinggera, P.; Franke, U.; Rother, C. Detecting Unexpected Obstacles for Self-Driving Cars: Fusing Deep Learning and Geometric Modeling. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Redondo Beach, CA, USA, 11–14 June 2017.
9. Luo, W.; Schwing, A.G.; Urtasun, R. Efficient Deep Learning for Stereo Matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
10. Schneemann, F.; Heinemann, P. Context-based Detection of Pedestrian Crossing Intention for Autonomous Driving in Urban Environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016.
11. Han, H.; Park, S. Traffic Information Service Model Considering Personal Driving Trajectories. *J. Inf. Process. Syst.* **2017**, *13*, 951–969. [CrossRef]
12. Kaiser, J.; Tieck, J.C.V.; Hubschneider, C.; Wolf, P.; Weber, M.; Hoff, M.; Friedrich, A.; Wojtasik, K.; Roennau, A.; Kohlhaas, R.; et al. Towards a Framework for End-to-end Control of a Simulated Vehicle with Spiking Neural

Networks. In Proceedings of the IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), San Francisco, CA, USA, 13–16 December 2016.
13. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to End Learning for Self-Driving Cars. *arXiv*, 2016.
14. Chen, C.; Seff, A.; Kornhauser, A.; Xiao, J. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015.
15. Pen, Y.; Cheng, C.A.; Saigol, K.; Lee, K.; Yan, X.; Theodoru, E.; Boost, B. Agile Off-Road Autonomous Driving Using End-to-End Deep Imitation Learning. *arXiv*, 2017.
16. Wang, S.; Clark, R.; Wen, H.; Trigoni, N. Deepvo: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
17. Yang, S.; Wang, W.; Liu, C.; Deng, W.; Hedrick, J.K. Feature Analysis and Selection for Training an End-to-End Autonomous Vehicle Controller using Deep Learning Approach. In Proceedings of the IEEE International Conference on Intelligent Vehicles Symposium (IV), Redondo Beach, CA, USA, 11–14 June 2017.
18. Lee, G.H.; Faundorfer, F.; Pollefeys, M. Motion Estimation for Self-Driving Cars with a Generalized Camera. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013.
19. Wolf, P.; Hubschneider, C.; Weber, M.; Bauer, A.; Härtl, J.; Dürr, F.; Zöllner, J.M. Learning How to Drive in a Real World Simulation with Deep Q-Networks. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017.
20. Lee, U.; Yoon, S.; Shim, H.; Vasseur, P.; Demonceaux, C. Local Path Planning in a Complex Environment for Self-Driving Car. In Proceedings of the IEEE 4th Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Hong Kong, China, 4–7 June 2014.
21. Canny, J.A. Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. [CrossRef]
22. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.
23. TORCS—The Open Racing Car Simulator. Available online: http://torcs.sourceforge.net/index.php (accessed on 4 January 2018).
24. TensorFlow. Available online: https://www.tensorflow.org/ (accessed on 4 January 2018).
25. OpenCV. Available online: https://opencv.org/ (accessed on 4 January 2018).