Learning a Deep Cascaded Neural Network for Multiple Motion Commands Prediction in Autonomous Driving

Xuemin Hu, Bo Tang, *Member, IEEE*, Long Chen, *Senior Member, IEEE*, Sheng Song, and Xiuchi Tong

***Abstract*— In autonomous driving, many learning-based methods for motion planing have been proposed in literature, which can predict motion commands directly from the sensory data of the environment, but these methods can neither predict multiple motion commands, such as steering angle, accelerator and brake, nor balance errors among different motion commands. In this paper, we propose a deep cascaded neural network for predicting multiple motion commands which can be trained in an end-to-end manner for autonomous driving. The proposed deep cascaded neural network consists of a convolutional neural network (CNN) and three long short-term memory (LSTM) units, fed with images from a front-facing camera installed at the vehicle. As the outputs, the proposed model can predict thee motion planning commands simultaneously including steering angle, acceleration, and brake to enable the autonomous driving. In order to balance errors among different motion commands and improve prediction accuracy, we propose a new network training algorithm, where three independent loss functions are designed to separately update the weights in the three LSTMs connected to three motion commands. We conduct comprehensive experiments using the data from a driving simulator and compare our method with the state-of-the-art methods. Simulation results demonstrate the proposed motion planning model achieves better accuracy performance than other models.**

***Index Terms*— Motion planning, autonomous driving, deep cascade neural network, motion command, LSTM.**

I. Introduction

AUTONOMOUS driving has received great interest from academia and industry [1], [2]. As one of the most important parts in autonomous driving, motion planning aims to make decisions for self-driving vehicles. Most of work focus on the solutions based on mathematical models restricted to specifically defined rules, which is called the rule-based method [3], [4]. The method of calculating planning results directly from perceived data is also called the end-to-end or learning-based method which requires a model that is able to obtain the relationships between perceived data and planning decisions. Compared with rule-based methods, learning-based methods have the ability of learning from the environment and do not need many time-consuming preprocessing stages. Moreover, inexpensive cameras are usually used in end-to-end methods, which can greatly decrease the cost of autonomous systems.

In the autonomous driving field, learning-based motion planning methods usually train deep neural networks using visual images from a front-facing camera paired with the time-synchronized motion commands which can be obtained from human drivers, computer games, or estimated by inertial measurement unit sensors [5]. These methods achieved good performance on steering for autonomous driving, but the accelerator and brake were not taken into account in most of these methods, whereas a vehicle needs to control not only the steering wheel but also the gas and brake when driving on a road. Thus, the methods which predict only one motion command can not meet satisfies of practical autonomous driving systems. Few methods consider both the steering angle and acceleration as motion commands, where a single loss function is used to evaluate the training errors of both steering angle and acceleration [6]. However, different motion commands require different prediction accuracies in autonomous driving. For example, a little change of the accelerator may not affect the driving state but a little change of steering angle may lead to an accident, especially when the vehicle speed is too fast. Additionally, human drivers may usually turn the steering wheel with a large range when turning a corner, but seldom fiercely hit the gas or the bake since intense driving is inadvisable in reality. Therefore, the methods of using only a single loss function to train all the output motion commands will lead to the problem that the accuracy of the predicted results is imbalanced, which affects the safety and stability of self-driving vehicles.

This paper aims to deal with the issue of predicting multiple motion commands for end-to-end autonomous driving, where the values of motion commands are calculated for the current frame according the current and previous frames. To learn

Manuscript received September 19, 2019; revised February 6, 2020 and April 28, 2020; accepted June 15, 2020. Date of publication July 3, 2020; date of current version November 29, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61806076, in part by the National Natural Science Foundation of China under Grant 61773414, and in part by Natural Science Foundation of Hubei Province under Grant 2018CFB158. The Associate Editor for this article was L. Li. *(Corresponding author: Long Chen.)*

Xuemin Hu and Xiuchi Tong are with the School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China (e-mail: huxuemin2012@hubu.edu.cn; 2431221584@qq.com).

Bo Tang is with the Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS 39762 USA (e-mail: tang@ece.msstate.edu).

Long Chen is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China (e-mail: chenl46@mail.sysu.edu.cn).

Sheng Song is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 200031, China (e-mail: 850250144@qq.com).

Digital Object Identifier 10.1109/TITS.2020.3004984

a deep planning model which can simultaneously predict multiple motion commands and improve prediction accuracy with balanced errors for different motion commands, a deep cascaded neural network for motion planning is proposed in this paper. In order to test the effect of simultaneously predict multiple commands in motion planning tasks, we consider the longitudinal accelerator as two separated commands, accelerator and brake. Moreover, three motion commands including the steering wheel angle, the accelerator and bake must be made by humans when driving a real vehicle, so the planning result in our method encompasses all the three motion commands instead of only the steering wheel angle [7], [8] or two commands including the steering angle and the longitudinal accelerator [6]. To interrelate the three motion commands, we associate them by using a shared convolutional neural network to extract spatial features from static images. However, these three commands are driven by different factors and their values are at different scales. For example, the steering angle is controlled according to bias of the heading angle in the movement, while the brake and throttle are controlled according the relative speed of obstacles. These temporal information can be extract by the long short-term memory, so three independent LSTMs are designed to handle the differences of the three commands. Therefore, in order to represent spatial-temporal features of the camera images and improve the planning precision, a novel deep cascaded neural network is developed, consisting of a CNN and three LSTMs which are individually connected to three output motion commands. We design three independent loss functions and employ them to separately update the weights of three LSTMs and the shared CNN. The training and testing data are collected from a driving simulator, with labels of three motion commands (i.e., simulated steering wheels, brakes and gas pedals) inputted by human players. The main difference between our method and the existing methods of combining a CNN and a LSTM is that multiple LSTMs can be connected after a shared CNN for outputting multiple independent commands that need to be decoupled, which can be used not only autonomous driving but also other motion planning tasks in automatic control fields.

In conclusion, our paper offers the following contributions:

- We propose a deep cascaded neural network consisting of a shared CNN and three independent LSTMs to handle the problem of simultaneously predicting the steering angle, the accelerator and brake value for autonomous driving, which is also a general structure for other multiple commands prediction in motion planning.
- A new neural network training algorithm is developed, where three independent loss functions are designed to separately update the weights of three LSTMs and the shared CNN in order to balance the errors among different outputs and improve the overall prediction accuracy.
- We collected driving videos for about ten hours from a driving simulator and tested our method in this dataset which we will publicly release in the near future. Experimental results show that our method outperforms state-of-the-art methods in the accuracy and smoothness performance.

The remainder of this paper is organized as follows: We review the related work in Section II. The details of the proposed approach are described in Section III, and experimental results in simulated traffic scenarios are given in Section IV. Finally, the conclusions are presented in Section V.

II. Related Work

As one of the most important parts in autonomous driving, planning has received a large amount of research interest. The research on planning can be divided into two primary categories: rule-based methods and learning-based methods.

Rule-based methods, including the A-Star, the D-star algorithms, the rapidly-exploring random tree (RRT) method, the discrete optimization-based method, and their variants, have made great achievements in the field of intelligent robots and autonomous driving vehicles [9]–[11]. However, several atomic and tractable tasks of recognizing relevant objects, such as road lanes, traffic signs and lights, pedestrians, etc., should be done by the perception system of autonomous driving vehicles before applying the rule-based planning methods [12]–[15]. Then, the recognition results are compiled to achieve a comprehensive understanding of the vehicle's immediate surroundings, and a safe and effective path or action can then be obtained through planning algorithms. These preprocessing stages before planning are usually time-consuming, which may lead to insufficient reaction time for the driver especially when an emergency occurs. In addition, some expensive sensors such as the LIDAR and the inertial navigation system are also used to detect objects and locate the vehicle position for planning algorithms in this process, which greatly increases the cost of an autonomous vehicle. Moreover, it is difficult to adapt traditional methods in scenarios beyond the rules since they are designed based on mathematical models restricted to specifically defined rules.

Learning-based methods gradually attract researchers' interest because of the advance of machine learning technologies in recent years. Compared with traditional methods, this kind of methods have the ability of learning from the environment and do not need many time-consuming preprocessing stages. In addition, inexpensive cameras are usually used in end-to-end methods, which can greatly decrease the cost of autonomous systems. These methods covers various research fields including mobile robots, autonomous driving, and unmanned aerial vehicle, etc., [16]. In these methods, the sensor data like visual images from vehicular cameras are directly fed into the planning model which is usually composed of neural networks, and the motion commands such as steering wheel angles are outputted after training the planning model using large number of training samples. According to the model architecture, learning-based methods can be mainly divided into two categories: reinforcement learning and imitation learning.

In reinforcement learning, hierarchical methods are usually used to construct multiple levels of temporally extended subpolicies [17]. The idea is to learn multi-purpose and parameterized controllers. Deisenroth proposed a method in which parameterized goals were used to train motion controllers in robotics [18]. A general framework for reinforcement learning with parameterized value functions, shared across states and goals was presented in [19]. Families of parameterized goals in the context of navigation was studied by Dosovitskiy and Koltun [20]. Sallab proposed a deep reinforcement learning framework for autonomous driving [21], and some researchers applied RL in end-to-end motion planning and predicted the steering angles for autonomous driving [22], [23]. It is not necessary to generate labels for training samples in the process of learning a planning model based on reinforcement learning, but the learning process needs interactive information between the agent and the environment. Since these methods usually perform poorly in the early learning stage, they are rarely used in practical autonomous driving systems for the safety reason.

Imitation learning has been applied in various tasks such as autonomous flight [24], articulated motion [25], and intelligent robots [26]. The input representations (raw sensory data or hand-crafted features), the predicted control commands, the learning model, and the learned representations are different for different applications. Most existing learning-based methods of motion planning for autonomous driving focus on imitation learning. In general, these methods train deep neural networks using visual images from a front-facing camera paired with the time-synchronized motion commands which can be obtained from human drivers, computer games, or estimated by inertial measurement unit sensors [5]. Convolutional neural networks are used to make robots learn from the environment and predict motion commands with perceived information, but only static obstacles are considered in this method. NVIDIA first proposed an end-to-end learning approach for self-driving cars based on a CNN [8]. The video frames were treated as individual images to learn a model to predict the motion commands, and the connections between the frames before and after the current frame were ignored in the two methods. A motion planning approach for steering autonomous vehicles considering temporal dependencies was developed to handle dynamic objects on roads [7]. Xu presented an end-to-end driving model learning from large-scale video datasets in [27]. Chi also proposed an end-to-end driving model to predict steering angles from spatial and temporal visual cues for autonomous driving [28]. These methods achieved good performance on steering for autonomous driving, in which the accelerator and bake were not taken into account, whereas a vehicle needs to control not only the steering wheel but also the gas and brake when driving on a road. Thus, these models which predict only one motion command can not meet satisfies of practical autonomous driving systems. Codevilla proposed a conditional imitating learning method for end-to-end driving in [6], where both the steering angle and acceleration are considered as the motion command. Nevertheless, a single loss function is used to evaluate the training errors of both steering angle and acceleration, which leads to the problem that the accuracy of two predicted motion commands is imbalanced.

We argue that human drivers controls a vehicle with different criterions for different commands. In this paper, we build an end-to-end learning model using a shared CNN and three independent LSTMs to extract both spatial and temporal visual cues from the driving scenes captured by a front-facing camera. Three individual loss functions are designed for outputting the driving commands of steering wheel angle, the accelerator and the bake, respectively, in order to balance errors from different motion commands and improve the prediction accuracy.

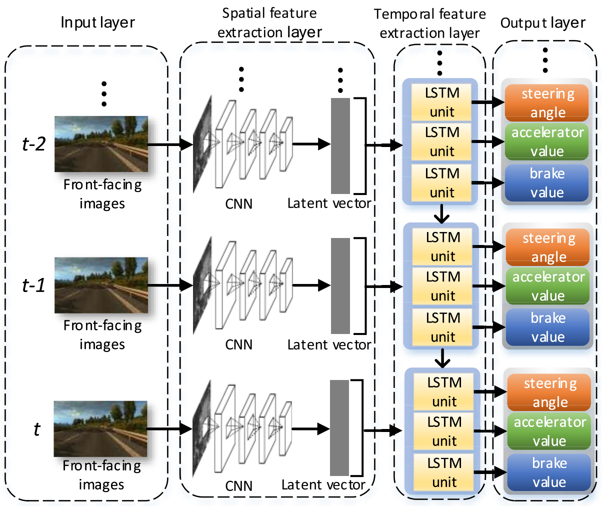

Fig. 1. Architecture of the proposed deep planning model with a shared CNN and three LSTMs.

III. Proposed Motion Planning Method

The architecture of the deep planning model is designed by a cascaded network combining a shared CNN and three LSTMs, which is shown in Fig. 1.

In order to clearly state the planning model, we conceptually segment the proposed network into four sub-networks including the input layer, the spatial feature extraction layer, the temporal feature extraction layer, and the output layer. The input images from the front-facing camera are first fed into the input layer where the images are normalized to the same size. The spatial feature extraction layer generates a fixed-length latent vector as the feature representation that succinctly models the visual surroundings and internal status of a vehicle. The compositive latent vector outputting from the spatial feature extraction layer are fed into three LSTMs in the temporal feature extraction layer, and the three LSTMs are connected to the three outputs representing the steering angle, the accelerator value and the brake value, respectively.

A. Input Layer

Human drivers mainly acquire information from the front scene by their eyes when driving a vehicle. Therefore, the input images with RGB channels at a time step are from a front-facing cameras, which are used to simulate the front scene captured by human drivers' eyes. The inputs of the proposed method are packed in a vector as (1).

$$\mathbf{p}_t = \{I_t^R, I_t^G, I_t^B\}, \tag{1}$$

where $\mathbf{p}_t$ is the perception at the time step t. I_t^R, I_t^G, and I_t^B represent the image channels of red, green, and blue, respectively, at the time step t. In this paper, the input image $\mathbf{p}_t \in \mathbb{R}^{224\times224\times3}$.

B. Spatial Feature Extraction Layer

Human drivers can observe road edges, lane lines, other vehicles, pedestrians, bicycles, traffic lights and signs, etc, and make rational decisions when driving a vehicle. CNNs such as AlexNet [29], VGG-Net [30], and ResNet [31] are effective models of expressing the features from observations. Considering the performance of feature extraction and the complication of networks, we choose a VGG-Net as our basis model. Among different VGG-Nets, the classical VGG-19 network has a balanced performance between feature extraction and complication in network architecture, so we design our spatial feature extract layer based on the VGG-19 network. In addition, the VGG-19 network is used to extract spatial features from the input images rather than classification in our method. Considering the network complexity and feature generalization, we get rid of the three FC layers and replace them with three convolutional layers where the kernel sizes are designed by 3×3, and the kernel numbers are 1024, 2048 and 4096, successively. Therefore, the output, as spatial features of the input image, is a latent vector with the size of $1 \times 1 \times 4096$.

Training a new deep network needs a large number of data and time. To obtain a good performance during a short time, the VGG-Net in our method is pre-trained on the Imagenet dataset that contains 1.2 million images of approximately 1000 different classes and allows for recognition of a generic set of features and a variety of objects with a high precision [32]. Then, we use our training data to make fine-turning and transfer the trained neural network from that broad domain to another specific one focusing on driving scene images.

C. Temporal Feature Extraction Layer

The VGG network can effectively extract spatial features from the images of driving scenes frame by frame, but human drivers take use of the information of previous frames in addition to the current frame. In other words, the correlation between frames should be used to predict the motion commands in our method Consecutive frames usually have similar spatial appearance, but subtle per pixel motions can be observed when optical flow is computed. Thus, traditional methods usually use the optical flow field to get the extracting temporal features between frames. In recent years, As improved Recurrent Neural Networks (RNN) models, LSTMs have shown successful on tasks of learning long-term dynamics, which provide a solution by incorporating memory units that explicitly allow the network to learn when to “forget” previous hidden states and when to update hidden states given new information [33]. We use the LSTM model as the subnetwork in our method, which was proposed by Donahue and used in the tasks of activity recognition, image captioning and video description [34].

As mentioned above, the spatial feature layer outputs a latent vector including 4096 feature maps with the size 1×1 as feature representations of the images from the vehicular camera at a time step. In the temporal feature extraction layer, the features before and after the current frames should be extracted and used to predict the values of the steering angle, accelerator and bake. In general, the three values have

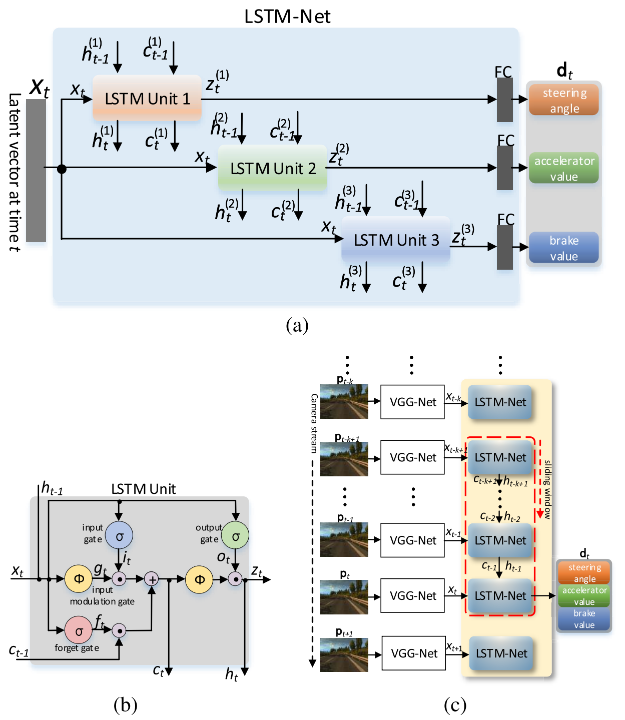

Fig. 2. Architecture of the temporal feature extraction layer. (a) LSTM-Net at time step t. The subscripts of x, h, c, and z represent the time, and the superscripts represent the index of LSTM units. (b) Details of each LSTM unit. Each LSTM unit includes a hidden unit h_t, an input gate i_t, an input modulation gate g_t, a forget gate f_t, an output gate o_t, and a memory cell c_t. x_t and z_t show the input and output of the LSTM-Net, respectively. (c) An overview of the LSTM network unrolled across time. The temporal feature extraction layer contains three LSTM units which are connected with the steering angle, the values of accelerator and brake with FC layers, respectively. Each frame from the front-facing camera is input into the VGG-Net, and the VGG-Net outputs the spatial features which are fed into the LSTM-Net. k successive frames are as a samples and the LSTM-Net outputs a group of motion commands.

different intervals, and the prediction accuracy of the steering angle is usually greater than the other values, because a little change of steering angle may lead to an accident when the vehicle speed is too fast. Additionally, human drivers may usually turn the steering wheel with a large range when turning a corner, but seldom fiercely hit the gas or the bake since intense driving is inadvisable in reality. Thus, the strength of controlling the steering wheel, the accelerator and the brake during a certain period of time is usually used as the criterion measuring whether a driver’s driving style is aggressive or not. In this case, the relationship between the change of the three different motion commands over time and the image features should be independently trained in an imitating learning model for autonomous driving systems. Therefore, We design the temporal feature extraction layer using three LSTMs to predict the steering angle, the accelerator value and the brake value, respectively, which is shown in Fig. 2.

Fig. 2(a) shows the architecture of the LSTM-net at time step t which includes three of the same LSTM units, and Fig. 2(b) shows the details of each LSTM unit. The LSTM unit in our method is derived from the version which was proposed in [34]. The latent vector outputted from the spatial feature extraction layer is fed into three LSTM units at each

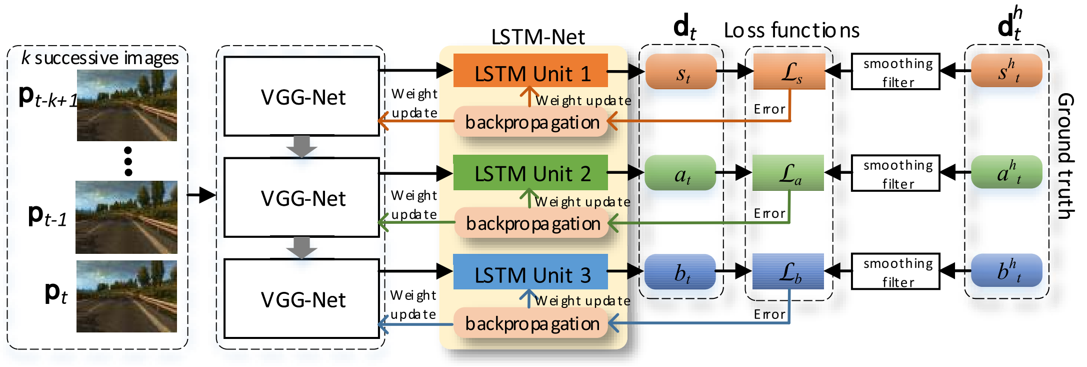

Fig. 3. Training process for the deep cascaded neural network.

time step t. Meanwhile, the LSTM unit at the time t receives the previous memory cell unit c_{t-1} modulated by f_t and g_t, and the previous hidden state h_{t-1} modulated by the input gate i_t. The values of i_t and f_t lie within the range [0,1] since they are sigmoidal. i_t and f_t can be thought of as knobs that the LSTM learns to selectively forget its previous memory or consider its current input. The output gate o_t is used to learn how much of memory cells to transfer to the hidden state. For predicting the steering angle, accelerator and bake, the output z_t of each LSTM unit is connected with a kind of motion command through a FC layer with the size of 256.

As shown in Fig. 2(c), the images from the vehicular camera are fed into the VGG-Net. The images from k successive frames are treated as a training sample. The LSTM-Net processes a sequence of the k fixed-length feature vectors (sliding window) extracted by the VGG-Net from the k successive images. The LSTM-Net learns to recognize temporal dependence leading to a planning result $\mathbf{d}_t$ based on the inputs from $\mathbf{p}_{t-k+1}$ to $\mathbf{p}_t$. A small value of k leads to faster reactions, but the network learns only short-term dependence. Whereas a large value of k results in a smoother behavior and more stable predictions, but increases the reaction time. In this paper, we set k as 10.

D. Output Layer

The aim of an end-to-end motion planning model is directly planning from perception. In this paper, planning results as the outputs of the deep network are motion commands for controlling the vehicle, including the steering angle, the accelerator value, and the brake value, which are packed in a vector as (2).

$$\mathbf{d}_t = \{s_t, a_t, b_t\}, \tag{2}$$

where $\mathbf{d}_t$ is the planning result at time step t. s_t, a_t, and b_t represent the steering angle, the accelerator value, and the brake value at time step t, respectively.

As shown in Fig. 2(a), the FC layer in the temporal feature extraction layer converts the output of the subnetwork to a 256-d vector, and the temporal feature extraction layer totally outputs three 256-d vectors. Each vector is connected with an output node which represents a kind of motion command, and the final output of the whole network is a 3-d vector. The value of the steering angle is normalized to the range [0, 2] in our method. The value 0 means turning the steering wheel to the end of the left, and the value 2 means turning the steering wheel to the end of the right. The value 1 means that the vehicle is travelling in a straight line. The values of the accelerator and bake are normalized to the range [0, 1].

E. Network Training Algorithm

As shown in Fig. 3, k successive images from the front-facing camera are collected as a training data and input into the network. Meanwhile, the synchronized motion commands given by human drivers are labeled as the ground truth. Since the motion command data given by human drivers usually contain some noise due to the control trembling, a smoothing filter [7] is used to reduce the noise of the steering angle, the values of accelerator and brake.

To predict three motion commands including the steering angle, the accelerator value and the brake value, a usual strategy is to design a weighted loss function to calculate the total loss and balance the training errors from different motion commands. However, it is difficult to determine the weights of the three motion commands in the loss function. A greater weight for the the steering angle means the training result is partial to the steering angle, and vice versa. In addition, it is not reasonable to set the weights as an equal value since the sensitivity requirement for the steering angle is usually greater than the accelerator and the brake for autonomous driving. In order to handle this issue, we proposed a training algorithm for the proposed deep cascaded neural network, where three independent loss functions is defined for the three predicted motion commands using the mean-square error (MSE) method, namely $\mathcal{L}_s$, $\mathcal{L}_a$, and $\mathcal{L}_b$, which can be formulated by (3), (4), and (5), respectively.

$$\mathcal{L}_s(\mathbf{w}_V, \mathbf{w}_s) = \mathbb{E}_{(\mathbf{p}_t, \mathbf{d}^h_t)^T_{t=1} \in \mathbb{D}}[\|s_t - s^h_t\|^2] \tag{3}$$

$$\mathcal{L}_a(\mathbf{w}_V, \mathbf{w}_a) = \mathbb{E}_{(\mathbf{p}_t, \mathbf{d}^h_t)^T_{t=1} \in \mathbb{D}}[\|a_t - a^h_t\|^2] \tag{4}$$

$$\mathcal{L}_b(\mathbf{w}_V, \mathbf{w}_b) = \mathbb{E}_{(\mathbf{p}_t, \mathbf{d}^h_t)^T_{t=1} \in \mathbb{D}}[\|b_t - b^h_t\|^2] \tag{5}$$

where $\mathbf{w}_V$ represents the weight vector in the VGG-Net. $\mathbf{w}_s$, $\mathbf{w}_a$ and $\mathbf{w}_b$ represent the weight vectors in the LSTM Unit 1, LSTM Unit 2, and LSTM Unit 3, respectively. These vectors are the weights to be optimized in the training process. $\mathbb{D}$ is the training data set with the labeled sequences $(\mathbf{p}_t, \mathbf{d}^h_t)^T_{t=1} \in \mathbb{D}$. $\mathbf{p}_t$, $\mathbf{d}^h_t$ and $\mathbf{d}_t$ are the perception, the synchronized motion

TABLE I
PARAMETERS OF IMPLEMENTATION DETAILS

parameter	value	parameter	value
mini-batches	4	input frame size	224×224
initial learning rate	0.0015	channel	3
iteration times	200,000	frames per second	30

commands (ground truth) and the planning results made by the model at time step t, respectively. The perception $\mathbf{p}_t$ which is shown by (1) is k successive sequential images at time step t. The ground truth $\mathbf{d}_t^h = \{s_t^h, a_t^h, b_t^h\}$ and the decision made by the model $\mathbf{d}_t = \{s_t, a_t, b_t\}$ both contains three motion commands including the steering angle, the accelerator and brake.

As shown in Fig. 3, the weights in each LSTM unit are separately updated in the training process. At the same time, the weights in the VGG-net need to be updated with the three LSTMs. Since The VGG-net in our method is designed to extract the spatial features which are general for all three outputted motion commands, the weights in the VGG-net will be partial to that commands after training if the weights in the VGG-net are updated only as the weights in the LSTM connected with any one of the three commands are updated. In order to balance the errors of the three motion commands and improve the overall prediction accuracy, the weights in the VGG-net are necessary to be updated as the weights in all of the three LSTMs are updated. For a batch training sample at time step t, the error between the predicted steering angle s_t and the corresponding ground truth s_t^h is calculated by the loss function $\mathcal{L}_s$ used to update the weights in the LSTM Unit 1 and the VGG-net with the backpropagation algorithm. Then, the error between the predicted accelerator value a_t and the corresponding ground truth a_t^h is calculated by the loss function $\mathcal{L}_a$ and used to update the weights in the LSTM Unit 2. Meanwhile the weights in the VGG-net are updated again. And then, the weights in the LSTM Unit 3 are updated and the weights in the VGG-net are updated once again. Thus, the weights in every LSTM unit are updated once, and the weights in the VGG-net are updated three times for a batch of samples. This process is shown by Alg. 1.

In this process, we use the Adam solver for backpropagation with mini-batches of 4 that means including 40 images since successive 10 frames are referred as an sample. Moreover, the learning rate is initially set to 0.0015, and it is multiplied by 0.9 in every 15,000 training steps. The parameters of implementation details are listed as Tab.I.

Algorithm 1 Training Algorithm for the Deep Cascaded Neural Network

```
Require: Training data set with the labeled sequences $(\mathbf{p}_t, \mathbf{d}_t^h)_{t=1}^T \in \mathbb{D}$.
Ensure: Parameters $\mathbf{w}_V$ in the VGG-net; Parameters $\mathbf{w}_s$ in the LSTM Unit 1 for the steering angle; Parameters $\mathbf{w}_a$ in the LSTM Unit 2 for the accelerator; Parameters $\mathbf{w}_b$ in the LSTM Unit 3 for the brake.
1: Choose the $\mathbf{w}_V$ after training on the Imagenet dataset;
2: Randomly initialize the parameters $\mathbf{w}_s$, $\mathbf{w}_a$, and $\mathbf{w}_b$ for the three LSTM units.
3: for Epoch = 1:N do
4:     Randomly sample a batch of sequential images with the synchronized commands $\{\mathbf{p}, \mathbf{d}^h\}$ from the training dataset $\mathbb{D}$.
5:     Calculate the loss function $\mathcal{L}_s(\mathbf{w}_V, \mathbf{w}_s)$ as (3).
6:     Update parameters $\mathbf{w}_V$ and $\mathbf{w}_s$: $(\mathbf{w}_V, \mathbf{w}_s) \leftarrow \arg\min_{\mathbf{w}_V, \mathbf{w}_s} \mathcal{L}_s(\mathbf{w}_V, \mathbf{w}_s)$.
7:     Calculate the loss function $\mathcal{L}_a(\mathbf{w}_V, \mathbf{w}_a)$ as (4).
8:     Update parameters $\mathbf{w}_V$ and $\mathbf{w}_a$: $(\mathbf{w}_V, \mathbf{w}_a) \leftarrow \arg\min_{\mathbf{w}_V, \mathbf{w}_a} \mathcal{L}_a(\mathbf{w}_V, \mathbf{w}_a)$.
9:     Calculate the loss function $\mathcal{L}_b(\mathbf{w}_V, \mathbf{w}_b)$ as (5).
10:    Update parameters $\mathbf{w}_V$ and $\mathbf{w}_b$: $(\mathbf{w}_V, \mathbf{w}_b) \leftarrow \arg\min_{\mathbf{w}_V, \mathbf{w}_b} \mathcal{L}_b(\mathbf{w}_V, \mathbf{w}_b)$.
11: end for
```

IV. SIMULATION RESULTS AND DISCUSSION

A. Experimental Environment and Evaluation Metrics

1) Dataset Description: To train a deep neural network, a large number of training data with corresponding labels should be first collected. On one hand, since existing public datasets for end-to-end autonomous driving do not contain sequential driving images with three synchronized motion commands including the steering angle, the accelerator value and the brake value, these datasets are not appropriate to apply in our experiments. On the other hand, it is a man-consuming and dangerous work to collect a mass of practical data of human drivers in various scenes, especially for scarce roads, weather and traffic scenarios. With the development of computer games, scenes in racing games are very similar to scenes in the real world. We argue that it can also perform well in the real world as long enough various training samples are provided if a model achieves a good performance in an artificial world. After assessing above aspects, we use synthetic data from computer games as the dataset in our method. Comprehensively considering the variety of traffic scenes, weather, and the area size, we select Euro Tuck Simulator 2 (ETS 2) as the experimental environment to collect the dataset in our method. We collected driving videos for about ten hours from ETS 2 and tested the proposed method. Some examples of the dataset are shown in http://github.com/hubudata/ETS_data.

2) Experimental Device: ETS 2 provides the perceived images in the front-facing view which we need in our method. Furthermore, we can obtain the steering angle, the values of the accelerator and brake through the input devices including the steering wheel, the accelerator pedal and the brake pedal when driving a vehicle in this simulator. When collecting training and testing data, we save 30 frames of driving images per second. Meanwhile, we collect the synchronized values of the steering angle, accelerator and brake for each frame from the input device.

The experiments are conducted on a personal computer with an Intel Core i7-7700K CPU (Quad-Core, 4.2GHz), 16GB×2 RAM and a NVIDIA GTX1080Ti GPU. All code is written in Caffe framework.

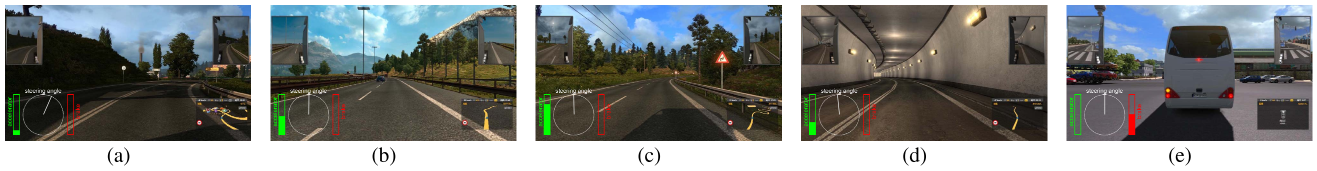

Fig. 4. Example images on different roads. (a) Country road, (b) Freeway, (c) Mountain road, (d) Tunnel, (e) Congested road. The white circle represents the steering angle, where the top center means 0, and the bottom center mean -1 or 1. The left side of the circle means 0 to -1 (turning to the left), and the right side of the circle means 0 to 1 (turning to the right).

3) Evaluation Metrics: In the processes of training and testing, the range of steering angle is [0, 2], and the ranges of accelerator and brake are [0, 1]. For uniformly analyzing and showing the experimental results, we normalize the steering angle s to the range $[-1, 1]$ before calculating the evaluation metrics in this paper. The commands given by the human driver are utilized as the criteria for evaluating the performance of the proposed model. Both Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) can represent average prediction error. Since large errors are particularly undesirable for the problem of motion command prediction, we use RMSE to report our results, defined as in (6) and give a relatively high weight to large errors.

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(d_i - d_i^h)^2}, \tag{6}$$

where d_i and d_i^h express the predicted motion commands from the proposed model and the ground-truth from human drivers for the frame i, respectively, in a testing dataset with a total of N frames. The variable d is replaced by the steering angle s, the accelerator value a, or the brake value b in this paper. MAE and RMSE which range from 0 to inf and are indifferent to the direction of errors, so they cannot infer how much stable motion commands are. Therefore, we refer to [7] and use another metric to measure the smoothness of the predicted motion command values, as shown in (7). Since the metric is used to measure smoothness, we use "SMO" as the variable name of the metric. The smaller the value of SMO is, the more smooth motion command variations are predicted.

$$SMO = \frac{1}{N}\sum_{i=1}^{N}(\frac{\partial d(t)}{\partial t}|_{t=i})^2, \tag{7}$$

B. Simulation Results and Performance Analysis

To further investigate the effectiveness of the proposed deep motion planning model, we report the performance from the following aspects.

1) Simulation Results on Different Roads: In order to test the performance of the proposed motion planning model, we collected amounts of videos from the ETS 2 combined with the synchronized motion commands as the ground truth, which mainly cover four roads including the country road, mountain road, freeway and tunnel. Since the measurements such as traffic states, turn signals, and the global position are not used in our method, urban roads with too many intersections are not considered in this paper. The proposed model was trained using these videos and performed iteration for 200,000 times. We additionally collected four videos as the testing dataset which includes each of the four types of roads. The road sections where we collected the testing dataset are not included in the training dataset. Since the traffic is light on the four testing roads, the brake is seldom used on these roads. In order to test the performance of predicting the brake value, a video of congested road is additionally collected as the fifth testing road.

Fig. 5 shows the testing results of predicting the steering angle on the country road, where the green and black curves show the results of the proposed motion planning model based on a CNN and three LSTMs with three independent loss functions (TILF) and the ground truth (GT), respectively. To visualize the testing results on the five roads, an example of each kind of road is shown in Fig. 4, where we present the prediction results on the game interface. Tabs. II and III show the results of $RMSE$ and SMO on the five testing roads, respectively. Thanks to the architecture with a CNN and three LSTMs, the proposed deep planning model can extract spatial and temporal features from the sequential images of driving scenes and effectively predict three motion commands including the steering angle, the accelerator value, and the brake value on various roads. In Fig. 5, the curves representing the results predicted by the proposed method and the results manipulated by the human driver are close to each other, which demonstrates the proposed method can simultaneously imitate three motion commands made by human drivers. The results in Tab. II demonstrate our proposed model performs well for predicting steering angles on all the roads except the tunnel. The $RMSE$ value of steering angle prediction tested on the country road is 0.0077, which is the smallest one among the five testing roads. However, the $RMSE$ value of accelerator prediction tested in the tunnel is the greatest one among all the testing roads. The reason is that the driving scenes are so similar that the image features are less evident in the tunnel than other roads. For predicting accelerator values, the proposed model perform better on the country road than the mountain road and tunnel since the mountain road and the tunnel contain many scenes with rises and falls whose features are obscure.

Since the traffic on the country road, the freeway, the mountain road and the tunnel is light, the brake is seldom used on the four roads. Thus, the values of brake prediction of the four testing roads are zero. To test the performance of brake value prediction, we collected a video from a congested

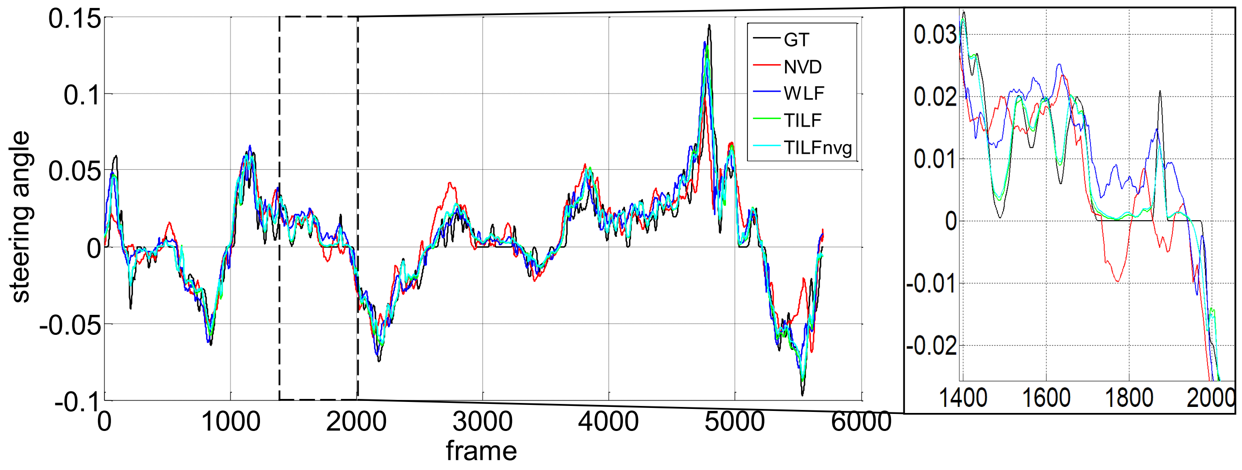

Fig. 5. Result of steering angle prediction on the country road.

TABLE II
COMPARED RESULTS OF $RMSE$ FOR THE FIVE TESTING ROADS

Road type	steering angle ($\times 10^{-3}$)			accelerator value ($\times 10^{-3}$)			brake value ($\times 10^{-3}$)		
	NVD	WLF	TILF	NVD	WLF	TILF	NVD	WLF	TILF
Country road	16.2	12.3	**7.7**	-	55.3	**33.4**	-	0	0
Freeway	23.6	19.4	**9.5**	-	63.0	**48.1**	-	0	0
Mountain road	31.2	16.4	**9.8**	-	98.8	**64.4**	-	0	0
Tunnel	35.5	26.1	**15.6**	-	78.4	**56.5**	-	0	0
Congested road	43.3	15.2	**8.6**	-	46.1	**32.5**	-	76.5	**42.0**

TABLE III
COMPARED RESULTS OF SMO FOR THE FIVE TESTING ROADS

Road type	steering angle ($\times 10^{-6}$)			accelerator value ($\times 10^{-6}$)			brake value ($\times 10^{-6}$)		
	NVD	WLF	TILF	NVD	WLF	TILF	NVD	WLF	TILF
Country road	**0.23**	0.37	0.31	-	12.78	**5.88**	-	0	0
Freeway	**0.17**	0.79	0.72	-	8.28	**2.58**	-	0	0
Mountain road	**0.75**	2.05	1.84	-	13.84	**6.72**	-	0	0
Tunnel	**0.30**	0.84	0.44	-	4.10	**1.47**	-	0	0
Congested road	**1.02**	1.83	1.60	-	36.43	**25.30**	-	35.33	**28.67**

road as an additional testing road, where many other vehicles are on the road and the ego-vehicle has to repetitively stop and go. The brake value prediction results are shown by the green curve in Tabs. II and III, in which the proposed model can effectively predict brake values as well as steering angles and accelerator values. As shown by an example in Fig. 4(e), the ego-vehicle avoids a collision with a dynamic obstacle by following it thanks to the designed network, where the image features of obstacles are extracted by the CNN layer and the moving features are extracted by the LSTM layer. In addition, training data contain lots of scenes that the ego-vehicles encounters dynamic obstacles and the synchronized motion commands made by human drivers, so the proposed model can learn to imitate human drivers' commands of avoiding obstacles. The result in Tab. II showing the $RMSE$ of the brake value tested on the congested road is 0.042, which means the planning results made by the proposed model are similar to the commands given by the human driver.

The smoothness performance of the proposed model is show in Tab. III which demonstrates that the SMO values of steering angle prediction tested on the mountain road are the greatest ones among all the testing roads because there are more turnings on this road, which leads to more changes of the steering angle for the human driver as well as the prediction results by the proposed method. The SMO performance of the accelerator and brake tested on the congested road is the worst one because there are always other vehicles in front of the ego-vehicle and the ego-vehicle has to repetitively go and stop. On the contrary, the proposed model performs better in the tunnel due to fewer turnings and ramps.

2) Comparative Results With Other Models: NVIDIA proposed a steering angle prediction method based on a designed CNN and achieved good results in lane following [8], which is one of the most classic end-to-end autonomous driving methods. However, it can only predict the steering angle. A deep network with a weighted loss function was developed to predict steering angles and accelerations in [6]. Existing methods do not predict the brake motion command, so, to the best of our knowledge, we are the first to simultaneously consider all the three motion commands for autonomous driving. Therefore, the method proposed by NVIDIA (NVD) is used as the comparative method for the steering angle prediction.

In addition, to demonstrate the advantages of predicting multiple commands of the proposed method, we make use of the idea proposed in [6] to construct a networks, a CNN and three LSTMs with a weighted loss function (WLF) for performance comparison in our experiments, where the weights of the three commands are set to 1, which means the three motion commands have equal importance. The comparative results are shown in Fig. 5, Tabs. II and III.

The curves in Fig. 5 and the data in Tab. II demonstrate the proposed TILF model has the best accuracy performance than the other two models for steering angle prediction. The NVD model can only predict steering angle values since a command output is designed in this model. Moreover, the NVD model does not perform as well as the other two models because the network in this model is too simple to express complex features of various kinds of roads. Especially, it performs worst on the congest road because the model is designed without LSTM network and can not handle the moving obstacles such as front vehicles. The WLF model has the ability of predicting the steering angle, the values of accelerator and brake, simultaneously, but the $RMSE$ values of this model are greater than the TILF models, which represents the accuracy performance of this model is worse compared with the TILF model. The reason is that only one weighted loss function is employed to calculate the errors from the three motion commands and update weights in the three LSTMs connected to the three motion commands. The errors from the three motion commands are different in the precision, and updating the weights in one LSTM will be affected by others, which leads to increasing the total training errors. On the contrary, a CNN and three LSTMs is used to extract spatial and temporal features from the sequential images in the proposed TILF model. In addition, three independent loss functions are designed and the weights in three LSTMs connecting the three motion commands are separately updated in the TILF model, which can balance errors of multiple different commands and improve the prediction accuracy.

The data in Tab.III show the comparative results of the smoothness performance for the three models. The NVD model has the smallest SMO values of the steering angle because it can not effectively predict the changes of the motion commands. The proposed TILF model has better smoothness performance than the WLF model since three LSTMs are designed in this model and the weights in these LSTMs are separately updated by three independent loss functions.

3) Discussion About the Compounding Error: The proposed method aims to predict multiple motion commands for autonomous driving using an imitation learning method, in which the compounding error is a common problem. The compounding error issue can be addressed within the proposed architecture by utilizing the idea developed in [8] with three cameras, including the left camera, the center camera, and the right camera, in the simulator when collecting training data. To demonstrate this, the center camera is the front-facing camera we mentioned in the previous sections, and two additional cameras are used to collect augmented training data that show the ego-vehicle in different shifts from the center of the current lane and yaw angles from the direction of the road. Images for two specific off-center shifts can be obtained from the left and the right cameras, and additional shifts between the cameras and all yaw angles are simulated by viewpoint transformation of the images from the nearest camera. The labels of steering angles, accelerator and brake values for transformed images are adjusted to those that would steer the ego-vehicle back to the desired location and orientation. A group of images from the three cameras are shown in Fig. 6. When on-line testing in the simulator, as mentioned in the previous sections, only the center cameras is used to collect input images. Through training the network in the augmented data, the proposed model can learn the ability of recovering from mistakes and preventing slowly drift off the road. Fig. 7 show some successive frames from game snapshots which demonstrate the ego-vehicle is trying to go through a right turn. Frames 1-4 show the ego-vehicle is navigating in the center of the current lane before entering the turn. Frames 5-8 show the ego-vehicle begins to turn right along the lane, but it deviates a little from the lane center because of compounding errors which can be seen from the right rear-view mirrors that are shown in the upper right corner of the images. Frames 9-12 demonstrate that the ego-vehicle adjusts the motion commands and goes back to the center of the current lane. The experimental results show the proposed method can deal with the compounding error problem by adding two additional cameras and augmenting the training data.

4) Discussion About the Influence by the Navigation Region in Driving Images: The navigation in driving images is very important to drive steadily along the predetermined route in our method, because the human driver can have a far view and estimate the road curvature thanks to the navigation. However, the navigation may cheat the network and affect the results of motion command prediction since it occupies a part of the driving image. In order to analyze the influence by the navigation region, we use the navigation region of the first frame to cover the navigation region of other frames in driving images so as to keep the navigation information unchanged, and then we conduct an additional comparative experiment in the country road. The RMSE values between the TILF method under the real-time changed navigation and the TILF method model under the unchanged navigation (TILFnvg) tested in the country road are shown in Tab.IV and and the result

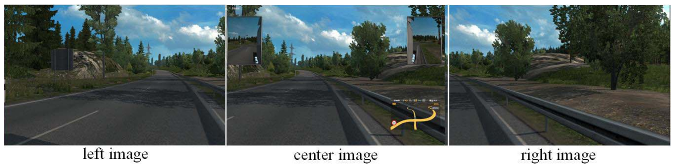

Fig. 6. Images from the left camera, the center camera, and the right camera.

TABLE IV

RMSE VALUES BETWEEN THE TILF METHOD UNDER THE REAL-TIME CHANGED NAVIGATION AND THE TILF METHOD UNDER THE UNCHANGED NAVIGATION IN THE COUNTRY ROAD

motion command	steering angle	accelerator value	brake value
RMSE ($\times 10^{-3}$)	0.55	2.12	0

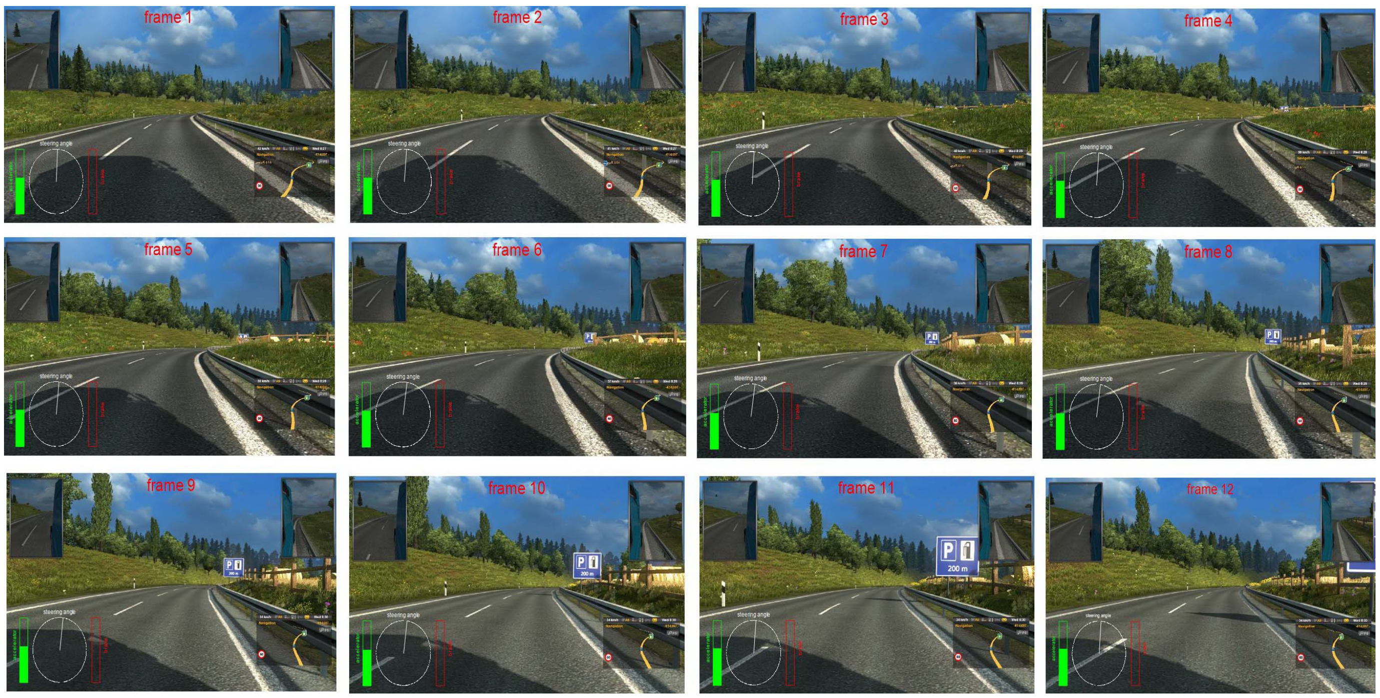

Fig. 7. Some snapshots of on-line testing in the simulator.

TABLE V

COMPARED RESULTS OF *RMSE* FOR DIFFERENT SAMPLING INTERVALS BASED ON THE TILF METHOD

Road type	steering angle ($\times 10^{-3}$)			accelerator value ($\times 10^{-3}$)			brake value ($\times 10^{-3}$)		
	SI=1	SI=3	SI=5	SI=1	SI=3	SI=5	SI=1	SI=3	SI=5
Country road	**7.7**	8.1	8.4	**33.4**	38.1	40.2	0	0	0
Freeway	**9.5**	15.0	17.5	**48.1**	59.9	64.2	0	0	0
Mountain road	**9.8**	10.1	10.2	64.4	**63.0**	65.7	0	0	0
Tunnel	15.6	**13.9**	14.2	56.5	53.8	**52.3**	0	0	0
Congested road	8.6	7.1	**6.5**	32.5	24.7	**20.4**	42.0	38.9	**33.2**

of the TILFnvg method is show in Fig.5. Both Tab.IV and Fig.5 demonstrate the results of motion command prediction of the TILF method under the unchanged navigation are very close compared with the results of the TILF method under the real-time changed navigation. It means the navigation indeed affects the prediction results, but the influence is very small. In addition, the navigation region locates in a small area (compared with the whole driving image) in the bottom right corner of the driving images rather than the center significant area, so the CNN usually pays little attention to this region in the process of the high level feature extraction. To visualize the feature maps extracted by the CNN layer, we project the feature maps of the last convolutional layer to the driving images by the method in [35]. Some examples of the projection are shown in Fig.8 which shows that the CNN pays most attention in the objects such as lanes, traffic signs, and other vehicles, etc., instead of the navigation region.

5) Discussion About the Sampling Interval of Frames Fed Into the Network: In our experiments, 30 frames of driving images per second are saved when collecting the training and testing data, and k successive frames are referred as a sample and input into the network, which means the sampling interval of frames fed into the network is set to 1. However, the hyper-parameter, sampling interval (SI), may affect the experimental results because different values of SI mean different lengths in temporal features of the driving video associated by the LSTM layer. In order to analyze the influence by the sampling interval, we conduct additional experiments with SI = 3 and SI = 5 based on the proposed TILF method, which means we sample one frame from 3 and 5 images of the collected successive driving images, respectively. In the comparative experiments, we retrain two models with the two sampling intervals, test them in the five tested roads, and compare the results with the original TILF method with SI = 1. The RMSE values of the results are shown in Tab.V,

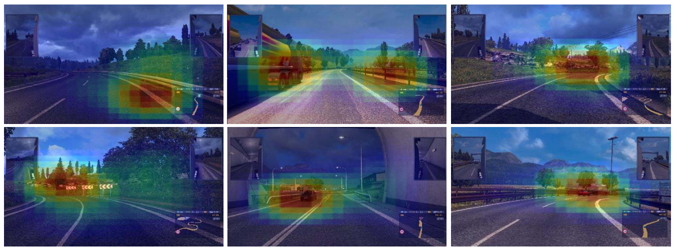

Fig. 8. Projection examples of the feature map of driving images.

which demonstrates that a larger sampling interval can obtain better results in the kinds of roads with a lower speed (e.g., the congested road) because the same k successive images with a larger sampling interval can get longer memories and remove redundant features between frames. Thus, the results with SI $= 5$ and SI $= 3$ are better than the results with SI $= 1$. However, if the vehicle navigates at a higher speed, a larger sampling interval will lead to miss some useful features between frames. Therefore, the results with SI $= 5$ and SI $= 3$ are worse than the results with SI $= 1$ when testing these methods in the freeway and the country road. The results tested in the mountain road and tunnel where the vehicle navigates at a middle speed are less affected by the factor of the sampling interval.

V. Conclusion

A deep cascaded neural network for multiple commands prediction in motion planning is proposed in this paper, which aims to address the problem of autonomously driving a vehicle solely from the vehicular cameras' visual observation. The sequential images captured from the front-facing camera are fed as the inputs to collect the information in front of the ego-vehicle. The proposed cascaded neural network consists of a shared CNN and three LSTMs. To extract the image features of driving scenes, the VGG-19 network is used as the spatial feature extraction layer. For expressing the correlation between front and back frames, we proposed a temporal feature extraction layer consisting of three independent LSTMs and employ it to predict the steering angle, the values and the accelerator and brake, respectively. In order to balance the loss errors for different outputs and improve the prediction accuracy, a network training algorithm is developed, where three independent loss functions are designed and used to separately update the weights in three LSTMs which are connected to three motion commands, respectively. The proposed motion planning method can simultaneously predict three motion commands including the steering angle, the values of the accelerator and the brake rather other only one command, while achieving competitive performance in prediction accuracy and smoothness compared with state-of-the-art methods. Future work will focus on reducing the complication in network architecture and adding other information such as the global navigation into our model. Furthermore, to improve the driving safety, we plan to integrate the data from multiple kinds of sensors including camera, lidar, and high precision map, etc., and collect training samples from real human driving records to test the performance of our approach in practical applications. In addition, using intermediate outputs of the network (e.g., attention features, semantic segmentation maps) to participate in calculating the training loss and improving the interpretability of end-to-end motion planning are also our future work.

References

[1] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C, Emerg. Technol.*, vol. 60, pp. 416–442, Nov. 2015.

[2] L. Chen, Q. Wang, X. Lu, D. Cao, and F.-Y. Wang, "Learning driving models from parallel end-to-end driving data set," *Proc. IEEE*, vol. 108, no. 2, pp. 262–273, Feb. 2020.

[3] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[4] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "Efficient sampling-based motion planning for on-road autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1961–1976, Aug. 2015.

[5] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2722–2730.

[6] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," 2017, *arXiv:1710.02410*. [Online]. Available: http://arxiv.org/abs/1710.02410

[7] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-end deep learning for steering autonomous vehicles considering temporal dependencies," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, pp. 1–8, 2017.

[8] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: http://arxiv.org/abs/1604.07316

[9] L. Zuo, Q. Guo, X. Xu, and H. Fu, "A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots," *Neurocomputing*, vol. 170, pp. 257–266, Dec. 2015.

[10] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mech. Syst. Signal Process.*, vol. 100, pp. 482–500, Feb. 2018.

[11] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, "A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic systems," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 6, pp. 2568–2578, Dec. 2018.

[12] H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2584–2601, Oct. 2017.

[13] I. Shim *et al.*, "An autonomous driving system for unknown environments using a unified map," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1999–2013, Aug. 2015.

[14] L. Chen, X. Hu, T. Xu, H. Kuang, and Q. Li, "Turn signal detection during nighttime by CNN detector and perceptual hashing tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3303–3314, Dec. 2017.

[15] L. Chen, W. Zhan, W. Tian, Y. He, and Q. Zou, "Deep integration: A multi-label architecture for road scene recognition," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4883–4898, Oct. 2019.

[16] Y. Chen, L. Liu, Z. Gong, and P. Zhong, "Learning CNN to pair UAV video image patches," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 12, pp. 5752–5768, Dec. 2017.

[17] Y. Jiang, J. Fan, T. Chai, J. Li, and F. L. Lewis, "Data-driven flotation industrial process operational optimal control based on reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 1974–1989, May 2018.

[18] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, "Multi-task policy search for robotics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 3876–3881.

[19] T. Schaul, H. Dan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1312–1320.

[20] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.

[21] A. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 2017, no. 19, pp. 70–76, Jan. 2017.

[22] M. Kaushik and K. M. Krishna, "Learning driving behaviors for automated cars in unstructured environments," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 583–599.

[23] Z. Ruiming, L. Chengju, and C. Qijun, "End-to-end control of kart agent with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2018, pp. 1688–1693.

[24] A. Giusti *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 661–667, Jul. 2016.

[25] P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth, "Model-based imitation learning by probabilistic trajectory matching," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 1922–1927.

[26] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1527–1533.

[27] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1–9.

[28] L. Chi and Y. Mu, "Deep steering: Learning end-to-end driving model from spatial and temporal visual cues," 2017, *arXiv:1708.03798*. [Online]. Available: http://arxiv.org/abs/1708.03798

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 1–9, Oct. 2017.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[32] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 677–691, Apr. 2017.

[35] M. Bojarski *et al.*, "VisualBackProp: Efficient visualization of CNNs," 2016, *arXiv:1611.05418*. [Online]. Available: http://arxiv.org/abs/1611.05418

Xuemin Hu received the B.S. degree in biomedical engineering from the Huazhong University of Science and Technology and the Ph.D. degree in signal and information processing from Wuhan University in 2007 and in 2012, respectively. Since July 2012, he has been an Associate Professor with the School of Computer Science and Information Engineering, Hubei University, Wuhan, China. He was a Visiting Scholar with the University of Rhode Island, Kingston, RI, USA, from November 2015 to May 2016. His areas of interest include motion planning, computer vision, and autonomous driving.

Bo Tang (Member, IEEE) received the B.S. degree from the Department of Information Physics, Central South University, Changsha, China, in 2007, the M.S. degree from the Institute of Electronics, Chinese Academy of Science, Beijing, China, in 2010, and the Ph.D. degree in electrical engineering from the University of Rhode Island in 2016. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS, USA. His research interests include statistical machine learning, computational intelligence, computer vision, and robotics.

Long Chen (Senior Member, IEEE) received the B.S. degree in communication engineering and the Ph.D. degree in signal and information processing from Wuhan University, Wuhan, China. From October 2010 to November 2012, he was a co-trained Ph.D. Student with the National University of Singapore. He is currently an Associate Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His areas of interest include autonomous driving, robotics, and artificial intelligence where he has contributed more than 100 publications. He received the IEEE Vehicular Technology Society 2018 Best Land Transportation Paper Award. He serves as an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE TRANSACTIONS ON INTELLIGENT VEHICLES, and the IEEE/CAA JOURNAL OF AUTOMATICA SINICA (JAS).

Sheng Song received the B.S. degree in electronic information engineering from Hubei University, China, in 2019. He is currently pursuing the M.S. degree in electronic engineering with ShanghaiTech University, China. He is interested in computer vision and medical image processing.

Xiuchi Tong received the B.S. degree in communication engineering from Hubei University, Wuhan, China, in 2018, where she is currently pursuing the master's degree with the School of Computer Science and Information Engineering. Her areas of interest include deep learning and motion planning.