End-to-end learning for high-precision lane keeping via multi-state model

Wei Yuan¹, Ming Yang¹, Hao Li², Chunxiang Wang¹, Bing Wang¹
¹Department of Automation, Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Ministry of Education of China, Shanghai 200240, People’s Republic of China
²Department of Automation & SJTU-ParisTech Elite Institute of Technology, Shanghai Jiao Tong University, Shanghai 200240, People’s Republic of China
✉ E-mail: MingYANG@sjtu.edu.cn

Abstract: High-precision lane keeping is essential for the future autonomous driving. However, due to the imbalanced and inaccurate datasets collected by human drivers, current end-to-end driving models have poor lane keeping the effect. To improve the precision of lane keeping, this study presents a novel multi-state model-based end-to-end lane keeping method. First, three driving states will be defined: going straight, turning right and turning left. Second, the finite-state machine (FSM) table as well as three kinds of training datasets will be generated based on the three driving states. Instead of collecting the dataset by human drivers, the accurate dataset will be collected by the high-performance path following controller. Third, three sets of parameters based on 3DCNN-LSTM model will be trained for going straight, turning left and turning right, which will be combined with FSM table to form a multi-state model. This study evaluates the multi-state model by testing it on five tracks and recording the lane keeping error. The result shows the multi-state model-based end-to-end method performs the higher precision of lane keeping than the traditional single end-to-end model.

1 Introduction
In recent years, several key technologies have achieved great progress in the self-driving field [1]. The end-to-end learning-based solution, which is an important part of the intelligence technology, has achieved significant progress because of the explosion of deep learning, enormous labelled datasets and more and more high-performance computing hardware such as field programmable gate array and graphics processing unit (GPU) [2].

The end-to-end learning-based autonomous driving approach is defined as driving using a self-contained system that maps from a sensory input such as front-facing camera images to directly perform driving actions such as accelerating, braking and steering. Therefore, the end-to-end model is self-optimised based on the training data and there are no manually defined rules in the end-to-end control process. The two major advantages of end-to-end learning are better performance and less manual effort. Since the model is self-optimised based on the data to give the maximum overall performance, the intermediate parameters are self-adjusted to be optimal. Moreover, the end-to-end system will be more efficient because there will be no need to divide the lane keeping driving system into several stages.

With the development of the autonomous driving and the improvement of road utilisation, the width of the road will be more and more narrow in the future. Hence, precision lane keeping is essential for the future autonomous driving. However, there are some challenging issues with the current state-of-the-art end-to-end lane keeping works. On one hand, the dataset for training an end-to-end model always contains imbalanced steering angle types, for example, 60% for going straight, 20% for turning left and 20% for turning right. As a result, a single end-to-end driving model always has a regularisation term to penalise too wide changes of the steering angle and more likely to predict straight behaviour. On the other hand, the datasets for training model is always collected by human drivers both in real world and simulators, which cannot guarantee the best accuracy of lane keeping. Thus, the current single end-to-end model will cause poor lane keeping effect in autonomous driving (see Fig. 1).

This paper presents a novel multi-state model-based end-to-end lane keeping method with a more accurate dataset collection strategy. On one hand, the dataset for training the end-to-end model will be collected by a high-performance path following controller. On the other hand, the multi-state model switched by finite-state machine (FSM) can predict widely steering angle and perform high-precision lane keeping the effect. The remainder of this paper is structured as follows: Section 2 will give related work in the field of end-to-end learning-based autonomous driving. Section 3 will describe the FSM for different driving states. Section 4 will introduce the details of the method including data pre-processing, the structure of the multi-state model, training and testing procedures and the evaluation method. Section 5 gives the experiments and the detailed analysis of the results. Section 6 will summarise the whole paper.

2 Related work
The pioneer work of end-to-end learning-based autonomous driving can be traced back to 1989. Dean (1989) trains a three-layer neural network to control the car to follow the road only by the inputs from a camera and a laser range finder [3]. This milestone brings a new way to achieve autonomous driving without decomposing the whole task into different modules. LeCun et al. (2004) map the raw input image to the steering angle based on a six-layer convolutional neural network for off-road mobile robots, which is the first application of CNN model in end-to-end learning-based autonomous driving [4]. Bojarski et al. [5] propose a CNN model, which contains five convolutional layers and three fully collected layers, to map the left image, the centre image and the right image from the front view to the steering angle. The end-to-end controller performs quite well after 3000 miles of learning. Yang et al. (2016) study the influences of features on the performance of training an end-to-end learning model by CNN [6]. The study demonstrates that the road-related features are indispensable for training a high-performance end-to-end control model.
Maqueda et al. [7] study the end-to-end steering prediction based on event-based vision, which uses the ResNet architecture.

The above end-to-end work maps the single input to the single output. However, the previous information will impact the current control in the real world. Therefore, some researches provide more reasonable models to predict sequence information. Xu et al. [8] propose a novel fully convolutional networks-long short-term memory (FCN-LSTM) architecture to train an end-to-end model based on large-scale video datasets. The novel FCN-LSTM end-to-end model takes a clip of video as input, which is more reasonable because the model takes historical information. Therefore, a 3D convolutional neural networks-long short-term memory (3DCNN-LSTM) end-to-end model will be designed as the basic end-to-end model in this paper. As analyses before, the single end-to-end model will have a regularisation term designed as the best threshold THL and THR will be searched according to the given step, which will make the category error smallest compared with the truth value of the road type. Fig. 3 gives a type splitting case.

According to the THL and THR, each steering value with responding state label will be defined as

\[ D_i = \{ (\delta, M_1), (\delta, M_2), \ldots, (\delta, M_n) \} \]  

Then, the FSM table will be generated by splitting the different steering values into three categories. In TORCS, each segment of the track has its own type, e.g. straight segment, left segment, and right segment, which will be the ground truth of generating FSM table. THL and THR are defined as the left edge and right edge of the three steering value categories. In Algorithm 1 (see Fig. 2), \( \text{step}_{\text{initial}} \) will be given first and the best threshold THL and THR will be searched according to the given step, which will make the category error smallest compared with the truth value of the road type. Fig. 3 gives a type splitting case.

According to Algorithm 1 (Fig. 2) will be 98.66% when set the \( \text{step}_{\text{initial}} \) to 0.001.

In the testing stage, different sets of training parameters will be switched by states, which is generating by comparing the current position with the FSM table. In the next section, the detailed end-to-end learning method will be given.

3 FSM based on driving states

The first procedure is to generate the FSM table. First, three steering control types are defined: the model of going straight, the model of turning right and the model of turning left. The three control types correspond to three different driving states

\[ s = \{ \text{straight}, \text{right turn}, \text{left turn} \} \]  

Second, a track and a car agent are selected in The Open Racing Car Simulator (TORCS) [22], which is the simulation platform for certificating our approach. Third, unlike traditional data collection strategy, this paper proposes a more accurate data collection strategy with high-performance path following controller. Stanley control model, a high-performance path following model, is applied for the car agent to follow the centre line of the track [23, 24]. Stanley model is defined as

\[ \delta = \theta_e + \tan^{-1} \left( \frac{u}{v} \right) \]  

\( e \) is the distance from the front wheel to the nearest point, \( v \) is the speed on the X-direction. The trajectory of the car agent and the centre line, the steering angle values and the front view image \( M \) along the whole track will be recorded. The raw training dataset will be defined as

\[ D_i = \{ (\delta, M_1), (\delta, M_2), \ldots, (\delta, M_n) \} \]  

1: \( \text{step} \leftarrow \text{step}_{\text{initial}} \)
2: \( \text{for} \, t_0 = 1 \rightarrow 1 + \text{step} \, \text{do} \)
3: \( \text{for} \, i = 1 \rightarrow n \, \text{do} \)
4: \( \text{Error}_{\text{type}} \leftarrow \text{Compare(type}(i), \text{type}_{\text{truth}}) \)
5: \( \text{end for} \)
6: \( \text{end for} \)
7: \( \text{THL}, \text{THR} \leftarrow \text{FindMinError}(t_1, t_n) \)

Fig. 2 Algorithm 1: Algorithm for generating FSM of road types

This is an open access article published by the IET, Chinese Association for Artificial Intelligence and Chongqing University of Technology under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)

Fig. 3 Road type of track chenyi-Wheel2. The type splitting accuracy by Algorithm 1 (Fig. 2) will be 98.66% when set the \( \text{step}_{\text{initial}} \) to 0.001.
4 End-to-end learning method

This paper will collect the front view images from TORCS by driver agent based on Stanley controller. Each image is labelled with ego-motion of the driver. Then, sequences of images with labelled steering angle values will be categorised to train different sets of parameters of the 3DCNN-LSTM model (i.e. parameters of going straight, parameters of turning left and parameters of turning right). At last, this paper will evaluate the multi-state model in TORCS by online driving. Fig. 4 shows the framework of the end-to-end learning with the structure of the multi-state model.

4.1 Data collection and augmentation

Chenyi-Wheel2 [25], a three-lane track, is selected as the basic track to collect the dataset. The original road surface textures of the track are replaced with customised asphalt textures and asphalt darkness levels. The agent driver is designed to follow the centre line with the reference speed of 35 km/h, the actual speed is around the 35 km/h. The training dataset is collected from only one track with the agent driver by 10 frames per second (FPS), as higher FPS can only serve to collect more similar images without providing useful information. The original screenshot will be cropped from 640×480 to 320×480 by cutting the top half of the image because the road-related features are indispensable for training an end-to-end controller [6]. Then, the labelled images are downsampled to 200×200. On the basis of FSM table, all the training samples will be categorised into three types and the single sample will be integrated into sequences samples. First, the sequence of samples eaten by 3DCNN is defined as

\[\text{eaten}_{ij} = \{(M, s)_{j1}, (M, s)_{j2}, \ldots, (M, s)_{js}\}\] (7)

Second, the sequence of samples which will be fitted is defined as

\[\text{seqfit}_{ij} = \{\delta, (M, s)_{j1}, \delta, (M, s)_{j2}, \ldots, \delta, (M, s)_{js}\}\] (8)

Third, one sequence of samples which will be fed into the model is defined as

\[\text{seq}_{ij} = \text{eaten}_{ij} \cup \text{seqfit}_{ij}\] (9)

At last, the whole dataset will be updated as

\[D_{\text{seqfit}} = \{\text{seq1}_{ij}^1, \text{seq1}_{ij}^2, \ldots, \text{seq1}_{ij}^m\}\] (10)

4.2 Structure of the 3DCNN-LSTM model

The structure of 3DCNN-LSTM model is shown in Fig. 4. The first part of the model is a 3DCNN structure, which takes the discrete time as the first dimension. The 3DCNN enables the model to learn motion detectors and understands the dynamics of driving. Specifically, the 3DCNN is composed of four 3D convolutional layers and four fully connected layers, where each layer will be a dropout in the training stage. The first 3DCNN layer has a kernel size of 3×12×12. The remaining three 3DCNN layers have a kernel size of 2×5×5. All the 3DCNN layers have 64 feature maps as output. Four fully connected layers have 1024, 512, 256 and 128 outputs. The second part of the model is the LSTM cell. The predicted angle serves as the input to the next time step. The loss is composed of two components: the mean square error (MSE) of the steering angle prediction in the autoregressive setting, the sum of MSEs for all outputs both in autoregressive and ground truth settings.

4.3 Structure of the multi-state model

Multi-state model is composed of 3DCNN-LSTM model based on three sets of trained parameters with FSM table, which is shown in Fig. 4. The current set of parameters of the model will be switched to another set of parameters by FSM table, which stores the coordinates of the whole track and will switch the state of the car by the coordinates. During the training stage, the dataset will be split into three categories according to data pre-process rule. Then, the three categories of the dataset will train the 3DCNN-LSTM model to produce three sets of parameters, e.g. Para S, Para L and Para R. During the testing stage, the multi-state model will be applied to control the agent car in TORCS. Therefore, the final prediction of steering value will be defined as steer_{ij} with regard to the multi-state model as a mapping

\[\text{steer}_{ij} = F(\text{seq}_{ij}, \text{pos}_s)\] (11)

pos_s is the current position, which will be used to compare with the FSM table and get the current driving state and responding set of parameters of the model, e.g. Para S, Para L and Para R.
4.4 Evaluation method

This paper will evaluate the performance of end-to-end controller by online driving. As shown in Fig. 4, TORCS will give the front view images and ego-motion information to the multi-state model by the shared memory in the testing phase. The generated steering value by the multi-state model will be given to the driving controller, which will generate the steering, the gear and the brake to the TORCS via the shared memory. The trajectory of the agent car, as well as the error to the centre line, can be recorded for each frame. The images and ego-motion information to the multi-state model by the feedback will generate the steering, the gear and the brake to the TORCS.

The multi-state model will be given to the driving controller, which will generate the steering, the gear and the brake to the TORCS. The lane keeping error $e_i$ is defined as the distance from the centre of the car and the nearest point on the centreline of the lane. Then, the root-MSE (RMSE) will be calculated as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} \quad (12)$$

To evaluate the degree of dispersion of the error, the standard deviation (SD) of all errors will be calculated as

$$ME = \frac{\sum_{i=1}^{n} e_i^2}{n} \quad (13)$$

$$SD = \sqrt{\frac{\sum_{i=1}^{n} (e_i - ME)^2}{n}} \quad (14)$$

5 Experiments and analysis

5.1 Training and validation

The training and validation will be conducted on Ubuntu14.04LTS with NVIDIA GTX 1080 GPU. After the data pre-process step, the 3DCNN-LSTM model is trained with the straight type of dataset, the left type of dataset, and the right type of dataset relatively. Each type of datasets contains 39,870 images (2658 sequences), 30,840 images (2056 sequences) and 23,355 images (1557 sequences). About 80% of each dataset is for training and 20% is for validation. The batch size will be set to four for the optimisation of the model. The learning rate of 0.0001. After each training procedure, validation will be conducted. The best performance of the model will be kept according to the validation results. During the training stage, the dropout will be set to 0.25, whereas in the validation stage the dropout will be set to 1. As Fig. 5 shows, the validation loss of straight-type parameter is stable after 200 epochs, whereas the validation loss of left and right type is stable after 600 epochs. Therefore, the whole training was stopped at 900 epochs.

5.2 Testing and evaluation of multi-state end-to-end model

This paper tests and evaluates the multi-state model in TORCS for online lane keeping. As shown in Fig. 6, five tracks are selected for online testing, e.g. chenyi-Wheel2, which is the track for data collection, another four tracks are similar to chenyi-Wheel2 but never be seen by the model. The same as the data collection stage, the reference velocity of the car is set to 35 km/h and the images from the TORCS will be downsampled by 10 Hz. To demonstrate the high performance of the multi-state end-to-end lane keeping model, a single 3DCNN-LSTM end-to-end model is trained with the whole dataset to compare with the multi-state model. Meanwhile, another single 3DCNN-LSTM model trained by dataset collected by the human driver is also used to implement the lane keeping. Since the steering value collected by the human driver cannot generate accurate FSM table, there is no need to train another multi-state model with such dataset. All the models will be evaluated in five tracks, the lane keeping time will be recorded and the RMSE and SD values will be calculated during the car keeping in the lane. The responding results are shown in Table 1.

On one hand, the lower value of RMSE means better lane keeping precision of the model. As shown in Table 1, Stanley control model owns the lowest RMSE value due to its high performance of path following, it finishes all the tracks. For end-to-end models, both multi-state model and single 3DCNN-LSTM model finish the track chenyi-Wheel2. Although they both fail to finish another four tracks, the multi-state model is much better than the single 3DCNN-LSTM model based on RMSE values on the whole. What is more, the multi-state model can keep the car in the lane much longer than the single model in track chenyi-e-track1. On the other hand, the lower value of SD means better stability of lane keeping. As demonstrated in Table 1, the multi-state model is much stable than the single end-to-end model. The single model trained with dataset collected by the human driver performs worst, which demonstrates that the accurate dataset collection strategy in this paper is much better than traditional human-driver-based dataset collection strategy.

To better illustrate the whole lane keeping precision, the track chenyi-Wheel2 and chenyi-Wheel1 are chosen to show the lane keeping precision. As shown in Figs. 7a and b, and the green line shows the smaller error produced by the multi-state model than the blue line produced by the single end-to-end model. Meanwhile,

![Fig. 5](image1.png)  
**Fig. 5** Loss of training and validation process

![Table 1](image2.png)  
**Table 1** RMSE and SD for the lane keeping error

![Fig. 6](image3.png)  
**Fig. 6** Illustration of tracks selected for real-time TORCS online control of lane keeping
Fig. 8 gives the three typical clips of trajectory, e.g. going straight, turning left and turning right. The trajectory produced by the multi-state model is much closer to the centre line of the lane. What is more, to illustrate the multi-state model can predict steering value more widely; this paper also applies the data collection strategy to chenyi-Wheel1 and gets the testing dataset. Then, Fig. 7c shows the prediction value based on both end-to-end models for the testing dataset. The regions A, B, C and D show that the multi-state model can predict prediction value more widely than the single end-to-end model. Although it is not the main evaluation method, it can demonstrate that this paper has solved regularisation term to penalise too wide changes of the steering angle for a single end-to-end control model. Fig. 9 shows some typical scenarios during the testing stage. The left top image is the current feeding image, the orange bar shows the current error between the current position of the car and the centre line, and the multi-state model will be switched by different states according to the position of the car. In short, the multi-state end-to-end model performs higher precision of lane keeping than the traditional single end-to-end model.

5.3 Analysis and discussion

Our target is to train general end-to-end model for high-precision lane keeping. Therefore, the lane keeping task is divided into three
different driving types, which are responding to different sets of training parameters based on one model. Our multi-state model is more general for different driving scenarios, which means different sets of trained parameters of a model are responsible for different pre-defined driving types, thus the model can predict wider steering value than the single end-to-end model. Meanwhile, the high-performance path following controller is applied to the data collection strategy, the data will be more accurate for training the model. In the real world, the dataset can be collected with the autonomous driving car following the centre line path based on the high-precision GPS. Then, the multi-state model trained by the dataset from the real world can be implemented with the help of the high-precision GPS.

However, this paper only defines three driving states of autonomous driving in the simulator. However, there are more driving states in the real world such as different weather conditions, light status and road textures. Therefore, more driving states need to be defined in real worlds such as rain-night-right-turn model etc. With the multi-state model, the more general end-to-end model can be trained by limited datasets and the whole end-to-end system will be more robust during autonomous driving.

6 Conclusion

This paper proposes a high-precision end-to-end lane keeping autonomous driving approach using multi-state model. A more accurate data collection strategy is applied based on the high-performance path following controller. Three sets of parameters of 3DCNN-LSTM end-to-end model are trained (i.e. going straight, turning right and turning left). This paper evaluates the multi-state model in TORCS for online driving to evaluate the lane keeping performance. The results demonstrate that the multi-state model performs higher lane keeping precision than the single end-to-end model. In the future, more driving states will be defined and the multi-state model will be applied in the real world.

7 Acknowledgments

This work was supported by the National Natural Science Foundation of China (U1764264/61873165). This manuscript is an extension of a conference paper presented at the 13th Chinese Conference on Intelligent Robotics.

8 References