Building an Autonomous Lane Keeping Simulator Using Real-World Data and End-to-End Learning

Zhilu Chen, Lening Li and Xinming Huang

This work was supported by the U.S. NSF under Grant CNS-1626236. *The authors are with Worcester Polytechnic Institute, Massachusetts 01609, USA. The corresponding author is Xinming Huang. E-mail: xhuang@wpi.edu*

xxxxxx

***Abstract*—Autonomous lane keeping is an important safety feature for intelligent vehicles. This paper presents a lane keeping simulator that is built with image projections of recorded data in conjunction with vehicle dynamics estimation. An end-to-end learning method using convolutional neural network (CNN) takes front-view camera data as input and produces the proper steering wheel angle to keep the vehicle in lane. A novel method of data augmentation is proposed using vehicle dynamic model and vehicle trajectory tracking, which can create additional training data as if a vehicle drives off-lane in any displacement and orientation. Experimental results demonstrate that the CNN model trained with the simulator can achieve higher accuracy for autonomous lane keeping and much lower failure rate. The simulator can serve as a platform for both training and evaluation of vision based autonomous driving algorithms. The experimental dataset is made available at http://computing.wpi.edu/dataset.html.**

Digital Object Identifier 10.1109/MITS.2018.2879224
Date of publication: xx.xx.xx

1939-1390/18©2018IEEE

I. Introduction

Lane keeping is a fundamental feature for intelligent and autonomous vehicles. Despite many sensors installed on autonomous cars such as radar, LiDAR, ultrasonic sensor and infrared cameras, the ordinary color cameras are still very popular owing to their low cost and ability to obtain rich information. Given the video images from the front-view camera, an vision based lane keeping system can automatically output the proper steering angles to keep the vehicle in lane. A traditional framework divides the task into several stages including lane detection [1], [2], path planning [3], [4] and control logic [5], [6]. Applying image processing techniques such as color enhancement, Hough transform and edge detection, the lane detection system is able to identify the lane markings on the road. Path planning and control logic are then employed to provide the proper steering angle adjustment for the vehicle. In this approach, performance of lane detection heavily relies on the feature extraction and interpretation of image data. Errors can also accumulate from a previous processing stage to the next, leaving the final control output less accurate.

In contrast, an end-to-end learning method has the advantages of better performance and less manual effort. End-to-end learning for self-driving cars has been successfully demonstrated in [7] using convolutional neural networks (CNNs), which takes the images from cameras as input and produce the vehicle control output automatically. The model is self-optimized based on the training data. The user does not need to manually define any features or rules or label the objects and their categories during the training process. Figure 1 is a comparison between the traditional framework and the end-to-end learning approach for vision based automatic lane keeping.

Although the approach of end-to-end learning for lane keeping is not new, the existing work has several deficiencies. For instance, the error difference between the recorded "ground truth" and predicted steering angle is not the best evaluation metric. Since it is hardly possible for a human driver to keep the vehicle perfectly in the center of the lane at all time, the recorded angles are not optimal. Thus, the predicted angles do not have to be exactly the same as the ground truth angles recorded from the human driving experience. It is more important to predict the position and orientation of the vehicle in the very next time step given the current vehicle speed and steering angle control. A more reliable evaluation metric can be provided by using a simulator, because the effects of the control input can be simulated and monitored.

Furthermore, we need to provide data to train the deep neural network to take appropriate steering angle actions when the vehicle drifts away from the center of the lane. However, the recorded driving data are lack of this type of actions since it is unsafe to drive off the lanes during data collection. To solve this dilemma, we propose a data augmentation method based on a vehicle dynamics model and vehicle trajectory tracking. Given any displacement and orientation, the model can generate a projected trajectory and a sequence of steering angle controls. Correspondingly, we can also create the augmented front views using image projection based on the displacements and orientation. Therefore, the system becomes a simulator that can not only generate augmented data for training the convolutional neural network, but also be used as a platform to evaluate the performance of other vision based lane keeping algorithms.

The main contributions of this paper are listed as follows:

1) This paper presents a simulator for vision based autonomous lane keeping. Although there are many recent works on lane keeping algorithms, it is hard to compare and evaluate them. Built on the recorded driving data, this simulator employs image projection, vehicle dynamics modeling, and vehicle trajectory tracking to predict vehicle movement and its corresponding camera views. The simulator can be used for both training and evaluation of lane keeping algorithms.
2) An end-to-end learning method is proposed that can generate proper steering angles from front-view camera data, which can keep the vehicle in lane. A highly effective end-to-end learning system is demonstrated using the aforementioned simulator. The CNN model trained with augmented data from the simulator performs significantly better than the model trained with recorded data only.
3) A completely new dataset for autonomous lane keeping is developed and was made available at http://computing.wpi.edu/dataset.html. The dataset contains recorded video frames from three forward facing cameras (left, center, and right) as well as steering wheel angles and vehicle speed information.

The rest of the paper is organized as follows. Section II summarizes related work on this topic. Section III provides the implementation details of our simulator, including image

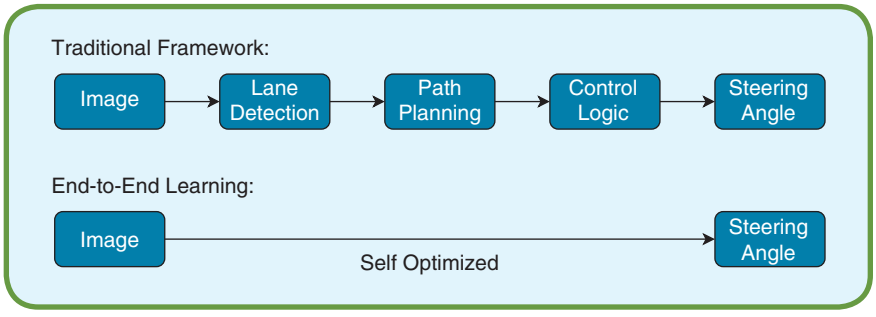


FIG 1 Comparison between the traditional framework and end-to-end learning for lane keeping.

projection, vehicle dynamics, vehicle trajectory tracking as well as the CNN architecture. The experiment and evaluation results are presented in Section IV, followed by discussions in Section V and conclusions in Section VI.

II. Related Work

Keeping the vehicle in lane is important for driving safety. Lane keeping assist systems (LKAS) have been studied by many researchers previously. Lane keeping assist systems [8]–[11] are able to provide torque to keep the vehicle within the lane, and often alert the driver with warning messages or sound. Cameras are usually used as the input. Lane marking recognition and road detection [12], [13] can be applied first to provide the target trajectory for vehicle control in a typical lane keeping assist systems. In addition, the systems also distinguish intended and unintended lane departure, by utilizing more information such as blinker state, braking or steering angle.

Lane keeping systems need to be accurate and robust for autonomous cars. Several recent research works on the theories, algorithms and implementations of lane keeping system were developed using virtual simulators. Deep reinforcement learning [14] was adopted for autonomous driving [15]–[17], in which the system could learn the optimal policy function from the feedback of reward. These systems went beyond the basic lane keeping feature, and were able to direct the vehicle to stay in path and avoid collision. The vehicle was not necessarily to be in lane, and other vehicles on the road were often involved. Learning and evaluation were often done in a virtual simulator, because learning requires rich ground truth information and needs to interact with the environment. Inverse reinforcement learning [18], on the other hand, was used to estimate the reward from the expert demonstrations.

For real world systems that do not have rich ground truth information as in a simulator, sensors and algorithms are employed to interpret the surrounding environment. The vision based approaches are popular since cameras are cost effective. An early research work was demonstrated on an autonomous vehicle named ALVINN [19], which uses a neural network to find the proper steering directions. The input came from a camera and a laser range finder, but the resolutions were quite small at that time. For high-resolution color images, an end-to-end learning approach using CNN was first presented by LeCun et al. [20]. The system was designed for off-road mobile robots, not for autonomous vehicles on the road. End-to-end learning using CNN for self-driving cars was demonstrated in [7]. It stated that a deep learning neural network was trained and evaluated using a simulator. The idea of building a simulator using image projection and vehicle dynamics was briefly mentioned, but there were little technical details. The network was later named as PilotNet, and the effectiveness was validated and visualized in [21], [22]. Our previous work [23] followed this approach using a different dataset and CNN model and achieved similar results for lane keeping.

Building a simulator requires the knowledge of computer vision, vehicle dynamics and vehicle trajectory tracking. Most of the existing automated driving frameworks have separate functional units for low-level motion control and path planning. Typically a nominal path can be obtained by optimization-based methods [24], sampling-based approaches [25] or other searching algorithms [26]. In terms of the system dynamics and control, Rami et al. [27] proposed a linear system dynamics and the control for high speed drifting. Galceran et al. [28] adopted proportional-derivative (PD) feedback controller for torque-based steering. Approximating the non-linearity of the vehicle dynamics, DeSantis et al. [29] Jacobian linearized the vehicle dynamics for designing a path-tracking controller, but this approximation ignored the high order of the polynomial of the system dynamics, which led to a potential problem of controlling a vehicle when the error is large.

III. Building a Simulator

A Overview

For evaluation of vision based lane keeping algorithms, a simulator is needed to provide feedback based on the predicted angle. The simulator can generate image frames to the vehicle position and orientation, and it can also simulate the vehicle movement giving a steer angle input. Therefore, a simulator for self-driving cars has two important components: graphic engine and physics engine. The graphic engine utilizes the information of the surrounding environment, as well as the pose of the camera to generate images. The physical engine simulates vehicle movement based on the input control actions. A virtual game engine usually contains both graphic and physics engine, and some autonomous driving simulators were built upon it [30], [31]. Vehicle movement simulation and frame generation can be integrated into the game engine. Besides, the ground truth information is very rich in the virtual world. Information such as vehicle position, orientation and velocity can be easily obtained, so do other objects. Despite these advantages, a significant drawback of these virtual simulators is that the generated images are still quite different from the real world data. Although they look very realistic with advanced graphic techniques, the details and variations of virtual images still cannot match the data from real world. It is risky to train a model using virtual game engines and then deploy the model for real-world driving. It would be better to build a simulator from the real world data.

Different camera views can be generated from recorded video frames by learning approach [32] or 3 D image projection approach [7]. The learning approach learns auto-encoders for embedding road frames, and learns

a transition model in the embedded space. The next few frames can be generated based on the current frame image and the current control inputs. On the other hand, the 3 D image projection approach assumes that the ground is a flat surface, and solves the 3 D geometry [33] to generate the next frame based on the actual recorded frame. The camera shift and rotation can be obtained from vehicle movement simulation, which can be estimated using vehicle kinematic or dynamic models [5], [6].

In our simulator, the image projection approach is employed for rendering the images. The CNN takes the image as input and the vehicle dynamics is used to simulate vehicle movement given the control action. Figure 2 shows the detail operations of the simulator when testing the CNN-based lane keeping algorithm. The predicted position is constantly validated against the ground truth position. A failure is recorded if the error exceeds a threshold value. More importantly, the simulator can be very useful when training the neural network by providing a large amount of additional training data through augmentation. When using the simulator for training, the vehicle trajectory tracking replaces the CNN controller to provide the control actions that can gradually correct initial position shift and/or orientation error. Practically, assuming an arbitrary shift and rotation of the vehicle from the ground truth, the vehicle trajectory tracking block can produce the proper steering angle control actions. Combined together with the generated camera view from the image projection process, augmented data can be generated. Figure 3 shows the operation flow of the simulator at training phase. During the training phase, many augmented data can be generated from each ground truth image by arbitrary shift and rotation of the vehicle.

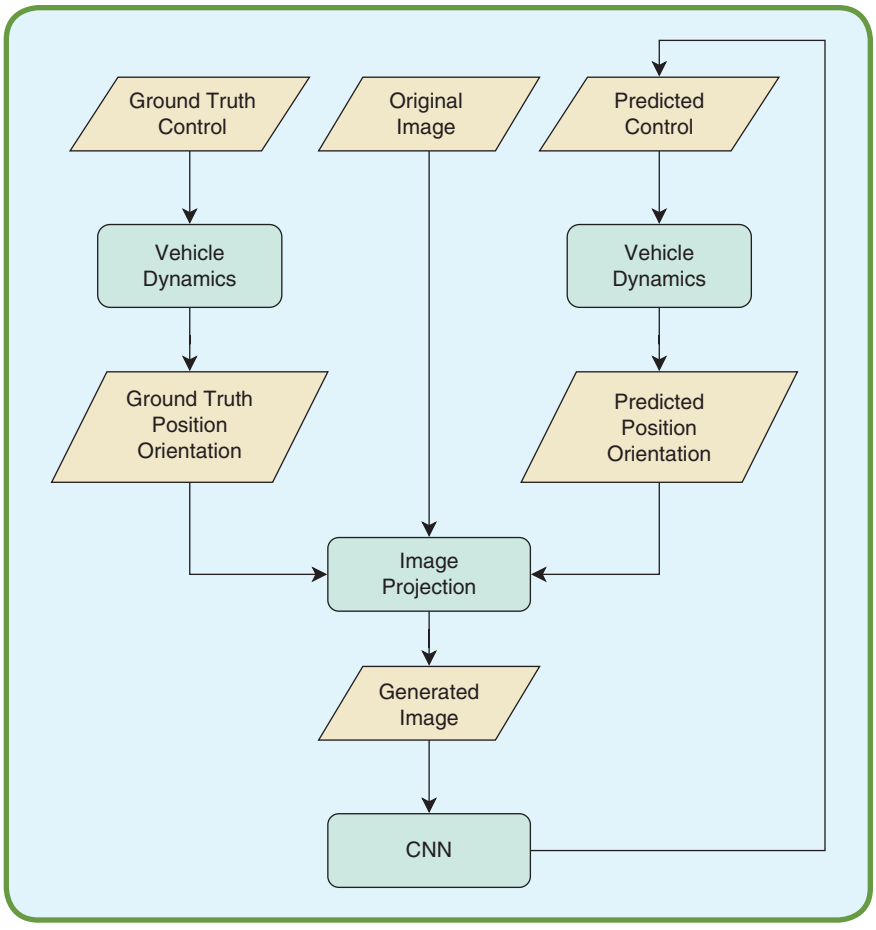


FIG 2 The simulator operation flow during the evaluation phase.

B. Image Projection

Rendering the image according to the vehicle position and orientation is required by the simulator, in order to provide more instances for machine learning and better evaluation metric. However, without using a gaming engine, data collected in real world are sparse, often along a single trajectory as the car goes. These data themselves are insufficient to cover all possible positions and orientations. Therefore, these data must be transformed for an arbitrary position and orientation, using computer vision knowledge of image projection base on 3 D geometry. Given a point in the world coordinates which is $X_w = (x_w, y_w, z_w)$ and the corresponding point in image coordinates which is $p = (p1, p2)$, there are relations that

$$\begin{aligned} p^h &= X_w^h M_{\text{ex}} M_{\text{in}} \\ X_w^h &= c(x_w, y_w, z_w, 1) \\ p^h &= d(p1, p2, 1) \end{aligned} \tag{1}$$

where p^h and X_w^h are 1×3 and 1×4 homogeneous coordinates, c and d are arbitrary nonzero constants, M_{ex} is the 4×3 extrinsic matrix and M_{in} is the 3×3 intrinsic matrix. Given the image taken in the real world with known calibration parameters M_{ex}, M_{in} and its pixels coordinates p, the new pixels coordinates $\tilde{p}$ need to be found with a new extrinsic matrix $\tilde{M}_{\text{ex}}$, when the camera is shifted and rotated. The physical dimensions of the 3 D scene are required in order to find the projection parameters. In the case of highway lane keeping simulation, it is assumed that the ground surface is flat, e.g., $z_w = 0$. The performance impact brought by this assumption is small,

according to the experimental results in [7]. According to (1), the mapping of p to $\tilde{p}$ then can be obtained as follows:

$$X_w^h = p^h M_{\text{in}}^{-1} M_{\text{ex}}^{-1} \tag{2}$$

$$\tilde{p}^h = X_w^h \tilde{M}_{\text{ex}} M_{\text{in}} \tag{3}$$

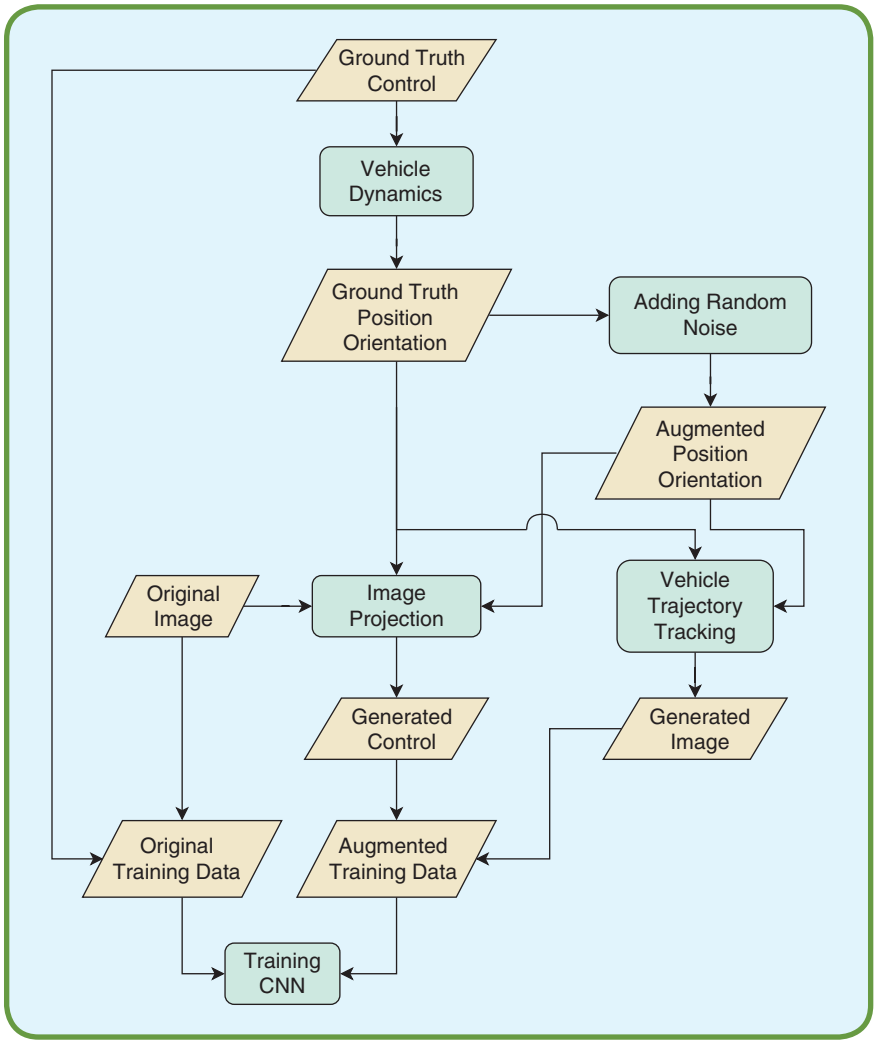


FIG 3 The simulator operation flow to generate augmented data during training phase.

Note that the lens distortion, if any, needs to be corrected before performing such image projection. Figure 4 shows some examples of transforming an original image according to camera's virtual position and orientation. The additive black area on the generated image is usually not an issue for vehicle simulation, since the captured images from front-view cameras are often cropped to retain only the middle section as the region of interest.

Another challenging task is ground surface estimation during calibration. To estimate the calibration parameters, especially M_{ex} in formula 1 with the assumption $z_w = 0$ for the ground surface, these three cameras used in our system need to be deployed on vehicle and world coordinates need to be established properly. When calibrating the cameras in the lab, a checkerboard pattern is usually used, as shown in Figure 4. However, estimating the ground surface needs a very large pattern, which is hard to craft and deploy. In our experiment, a flat parking lot with existing markings is used for ground surface estimation. Physical dimensions of the markings are measured manually while the corresponding images are captured by the cameras installed on the vehicle. Figure 5 shows the selected points in the image taken by the center camera during the calibration. The physical locations of the cameras and the selected points in the world coordinates are also shown in Figure 5. Three cameras are installed on the left, center and right of the vehicle, all facing forward, because they can provide better field of view than

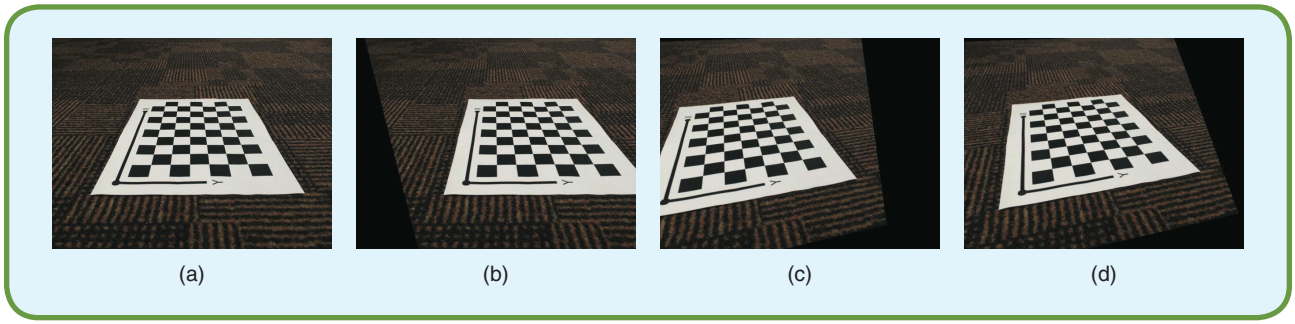
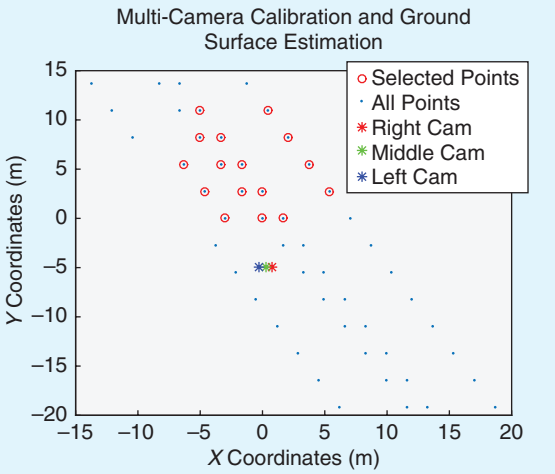


FIG 4 Example of original image and generated images given arbitrary camera poses. (a) Original image. A checkerboard pattern on a flat surface. (b) Generated image as if the camera is shifted left by 50 mm. (c) Generated image as if the camera is rotated right by 15.25 degrees. (d) Generated image as if the camera is shifted left by 50 mm and rotated right by 15.25 degrees.

(a)



(b)

FIG 5 Camera calibration and ground surface estimation. (a) Selected points in the image taken by the center camera. (b) Cameras and selected points in the world coordinates.

a single camera. In fact, the nearest camera to the vehicle's virtual position is selected as the source in equations 2 and 3. Therefore, the generated images have better quality and less additive black areas after projection.

C. Vehicle Dynamics and Vehicle Trajectory Tracking

According to [5], the bicycle vehicle dynamics shown in Figure 6 is captured by the following equations:

$$\begin{aligned}
\dot{x} &= v\cos\theta \\
\dot{y} &= v\sin\theta \\
\dot{\theta} &= \omega \\
\theta &= \psi + \beta \\
\dot{\psi} &= \frac{v}{l_r}\sin\beta \\
\dot{v} &= a \\
\beta &= \arctan\left(\frac{l_r}{l_f + l_r}\tan(\sigma_f)\right)
\end{aligned}$$

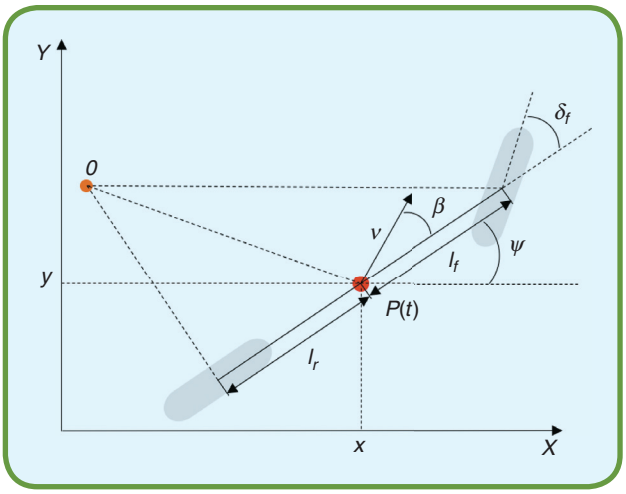


FIG 6 An illustration of virtual bicycle model for vehicle dynamics.

where $P = [x, y, \theta] \in \mathbb{R}^2 \times \mathbb{S}^1$ is the state of the position and orientation, v and ω are the linear velocity and angular velocity respectively, and they are assumed to be the control signals for the vehicle. a is the acceleration, and σ_f is the turning angle. l_f and l_r are the distance from the vehicle's mass center to the front and rear axles. In our test vehicle, we use estimated values $l_f = 1$ m and $l_r = 1.7$ m.

A nonlinear feedback mapping called feedback linearization is introduced upon the dynamics depicted in Figure 6 from the current nonlinear system to a new linear system and a new state variable $z = [x, y, \dot{x}, \dot{y}]$:

$$\dot{z} = Az + Bu$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = A \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} + Bu$$

where the state matrix $A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, the input matrix $B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$, and the input vector $u = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}$ in the new linear system. After the feedback linearization, the whole problem is transformed into designing the proper gain K for the linear system. To solve this optimal control problem, Linear Quadratic Regulator (LQR) is adopted to acquire the optimal gain K. The quadratic cost is defined as the following, which aims to minimize the error between the actual trajectory and the predicted trajectory:

$$J = \int_0^\infty (e^\top Q e + u^\top R u)\, dt \tag{4}$$

where $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and $R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, e is the error between the actual state and the predicted state, and u is the control effort. Practically, Q and R matrices do not have to be identity matrices but positive definite, and the entries can be tuned to achieve required performance accordingly. Once the gain K is computed, the feedback control law and the ordinary differential equation (ODE) of the new linear system are described as follows:

$$\begin{aligned} e &= z - z_d \\ u &= -Ke + u_d \\ \dot{e} &= (A - BK)\, e \\ \dot{z} &= \dot{z}_d + \dot{e} \end{aligned}$$

where e is the error between the real state and desired state, K is the gain computed based on the defined cost equation 4 with the corresponding A and B matrices, $u \in \mathbb{R}^2$ is the input vector, $u_d = (\ddot{x}_d, \ddot{y}_d)$ is the referenced input given by the ground truth. $\dot{e}, \dot{z}, \dot{z}_d$ are changing of the error, state, desired state, respectively.

$$v = \dot{x}\cos\theta + \dot{y}\sin\theta \tag{5}$$

$$\omega = \frac{1}{v}(\ddot{y}\cos\theta - \ddot{x}\sin\theta) \tag{6}$$

The control input for the nonlinear system can then be calculated by remapping the new input variables of linear system back to the original input of the nonlinear system shown in equations 5 and 6, which are linear velocity v and angular velocity ω.

The results in Figure 7 demonstrate the effectiveness and correctness of the vehicle trajectory tracking controller design. A vehicle with feedback control law has the capability of converging to and following the desired trajectory, even though there exists an initial error. At the beginning, owing to some errors between the predicted and actual orientations, the steering angle is positive and large, which helps the vehicle to correct its orientation in a short time. After 2 seconds, the predicted orientation converges to the ground truth. The vehicle orientation does not change rapidly for the next few seconds, which matches the fact that the steering angle of the vehicle remains in a very small range near zero.

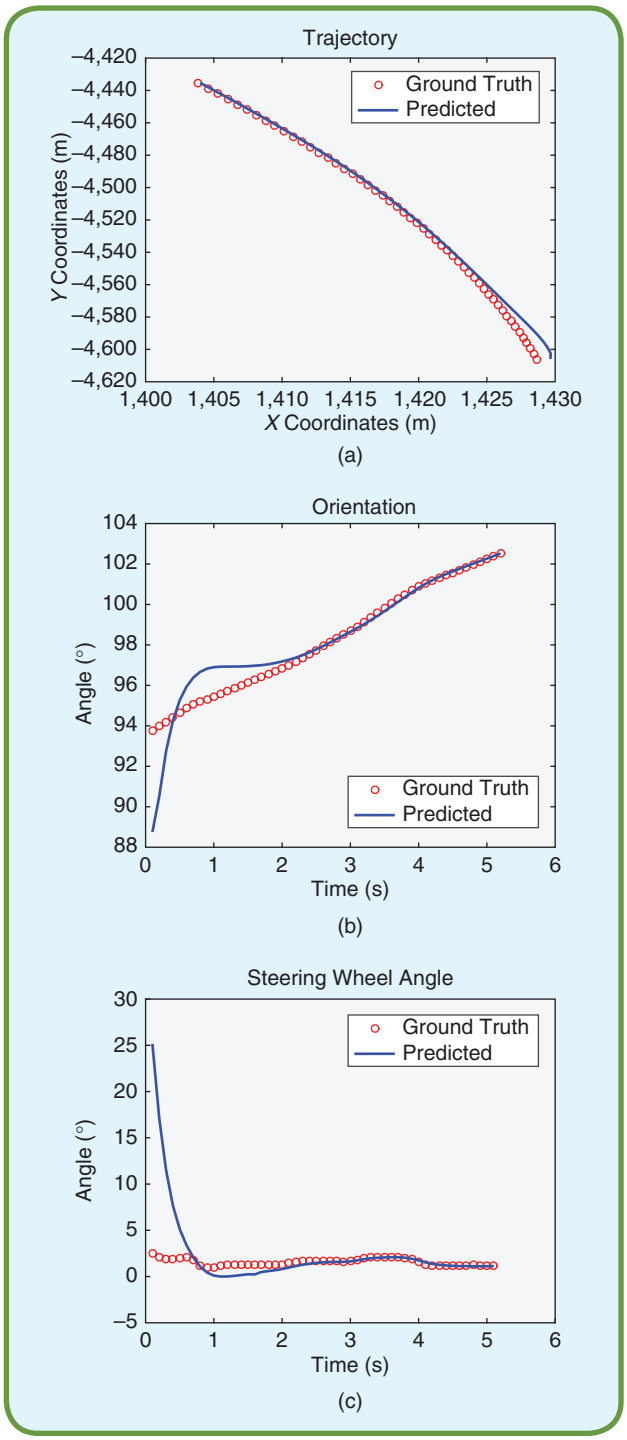


FIG 7 Correction of vehicle's position and orientation using vehicle trajectory tracking. (a) Ground truth and predicted trajectory. (b) Ground truth and predicted orientation. (c) Ground truth and predicted steering wheel angle.

D. CNN Implementation

Convolutional neural networks (CNNs) [34]–[36] has achieved impressive performance in image classification. In this paper, learning the human driver's control is not a classification problem but a regression problem, therefore the loss layer during training is Euclidean loss, which computes the sum of squares of differences between predicted steering angle and ground truth steering angle: $(1/2N)\sum_{i=1}^{N} \| x_i^1 - x_i^2 \|_2^2$, where N is the number of instances, x_i^1 is the ith predicted value and x_i^2 is the ith ground truth value. The CNN is used as a steering angle predictor given the input image. It does not take the entire image frame as input since only the center section is the region of interest for lane keeping. The images are cropped before fed

to CNN, as shown in Figure 8. The proposed CNN architecture is shown in Figure 9, and it is based on the PilotNet [7], [22]. It has 5 convolutional layers and 3 fully-connected layers. There are no pooling layers because the feature maps are small. The convolutional layers are mainly for feature extraction and the fully connected layers are mainly for steering angle prediction, but there is no clear boundary between them since the model is trained end-to-end. Unlike the PilotNet, our input image size is 400×150 instead of 200×66. The first convolutional layer is 4×4 stride and 9×9 kernel instead of 2×2 stride and 5×5 kernel. The system of PilotNet uses the vehicle's turning radius r as steering command, and makes the inverse-turning-radius $1/r$ as the output to avoid infinite numbers when driving straight. Our CNN uses the steering wheel angle as the output, which is more intuitive. The proposed CNN model is trained using our own dataset on Caffe [37] and Matlab software platform.

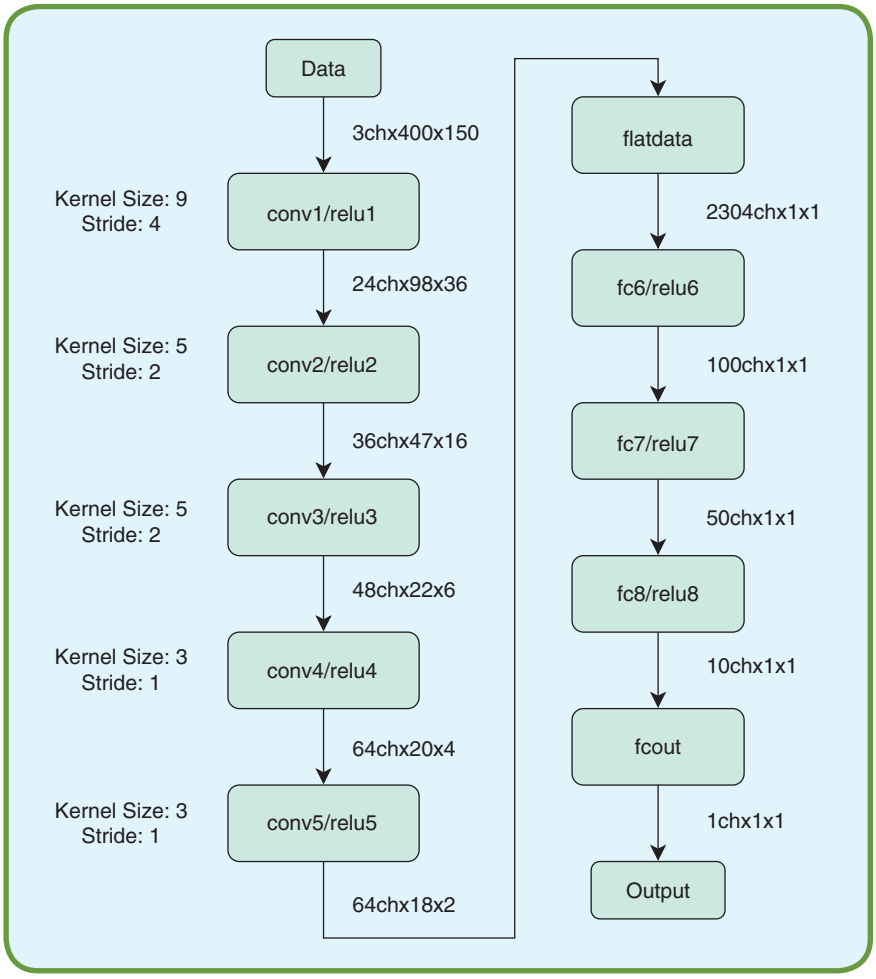


FIG 9 Architecture of the CNN for end-to-end learning.

IV. Experiment

A. Data Collection

To capture images, three forward facing cameras are mounted on the dashboard of the car, from left to right. Because the cameras are not water-proof, installing them on top of the vehicle can be inappropriate. To avoid re-calibration each time, the cameras remain stationary once installed. Multi-thread programming and software triggers are used to synchronize the three cameras to capture images at 10 Hz. The shutter time is set to auto with an upper-bound value to avoid extremely low frame rate when the light condition is too dark. The image resolution is set to 1288×968, and captured images are stored as color image sequences. Meanwhile, the steering angle and speed information are recorded by accessing the CAN BUS via OBD-II port. The data from OBD-II port are decoded by our customized program, and then

FIG 8 An example of cropped image frame from the dataset.

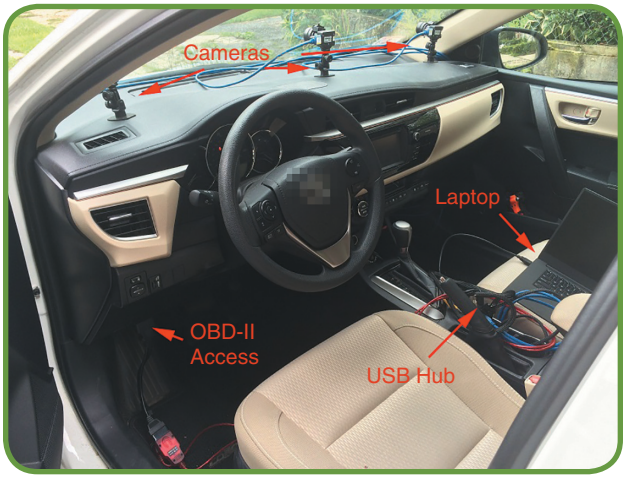


FIG 10 The data collection system including three forward facing cameras, a USB hub, a laptop computer with access to OBD-II port.

FIG 11 Example frames under different weather or lighting condition: (a) cloudy (b) shadowed (c) foggy (d) sunny.

saved with time stamps, in order to synchronize with the image data. The steering wheel angle decoded from the OBD-II port has a precision of 0.07 degree and the speed data has precision of 1 km/h or approximately 0.28 m/s. The steering wheel angle s need to be converted to vehicle's turning angle σ_f in Figure 6 by dividing the steering ratio k as $\sigma_f = s/k$, where k has an estimated value of 17.8 in our experiment. Figure 10 shows our data collection system on a vehicle, including three forward facing cameras, a USB hub, a laptop computer and an interface to OBD-II port.

The experimental data were collected on 7 occasions at 6 different days, approximately 1 hour each. Different lighting and weather conditions are included, such as sunny, cloudy and foggy, as shown in Figure 11. Night time driving is not included in our data. The collected data are then refined to be used for the task of lane keeping. Some recorded data that meet the following criteria are discarded: non-highway driving, speed lower then 40 mph, change of lane, extreme lighting condition, equipment failure, and sequences that are shorter than 1 minute. After refinement, about 3 hours of driving data are valid. Among the 7 groups of collected data, 4 groups were used for training and the other 3 groups are for testing. This is to prevent overlaps between training and test data. Overall, the train data contain 68082 frames, nearly 2 hours at 10 Hz. The test data contain 32053 frames, nearly 1 hour at 10 Hz. The training data sequences are randomly shuffled before applied to the CNN model.

B. Data Augmentation

Ideally, the training dataset should contain some error correction scenario such that the trained CNN model is capable of handling errors. So the vehicle stays in the lane instead of drifting away. Such error correction data introduce initial errors for the vehicle's position and/or orientation, and then provide the proper control action to correct such errors and guide the vehicle back to the lanes. The original data collected from highway driving are lack of such error correction data, because of the safety concern to perform such dangerous maneuvers on highway. Therefore, we propose to apply data augmentation technique that can generate this type of error correction data virtually. This is one of the important benefits of building a simulator.

Once the data are collected and the world coordinates established, it is possible to obtain the ground truth of vehicle's position and orientation at any given time. For each frame, errors can be added manually into the vehicle's position and orientation. By using the knowledge of image projection of 3 D geometry, the augmented images can be generated accordingly. At the same time, correct control action is provided by the vehicle trajectory tracking algorithm. Therefore, the augmented data can be used as part of the training data to improve the model's robustness. In our experiment, the simulator generates 10 augmented data from each image frame by randomly shifting the vehicle positions in the range of $[-1, +1]$ meter off the center and randomly changing the orientation in the range of $[-6, +6]$ degrees. Figure 3 shows the entire process of data augmentation. Figure 12 shows examples of augmented images.

C. Evaluation Using Simulator

In our previous work [23], it is shown that the differences of ground truth angle and predicted angle is not an effective metric for evaluating the performance of lane keeping systems. Hereby we propose a new metric by measuring the percentage of driving time when the vehicle is in lane. Our simulator can be employed as an evaluation platform for autonomous lane keeping. The process flow of using the simulator for evaluation is illustrated in Figure 2.

Giving the initial steering angle provided by the CNN model, vehicle positions and orientations are updated by vehicle dynamics. Subsequently a front-view camera image is generated through image projection according to the current vehicle position and orientation. The new image is then fed to the CNN model and it produces the steering angle for the next time-step. The same process repeats for all frames in a test sequence. At each time step, the amount of position difference to the ground truth is calculated. For the purpose of simplicity, the longitude difference is fixed to zero, and the lateral difference is

compared with a threshold value. If the lateral difference is larger than the threshold, it is considered a lane keeping failure. The threshold is set to 1 meter in our experiment, the same as in [7]. In our dataset, the human driver cannot guarantee that the vehicle travels exactly following the center trajectory of the lane. Nevertheless, the recorded position is assumed to be lane center. Therefore, we choose a larger threshold to avoid the ambiguity of lane departure. A smaller threshold could be too restrictive for model evaluation using this dataset. For each failure occurrence, the next 60 frames are automatically marked as manual driving period. All other frames without failure are considered autonomous driving period. The final criteria is the percentage of autonomous driving time (autonomy):

$$A = \frac{t_a}{t_a + t_m} \tag{7}$$

where t_a and t_m represent the autonomous time and manual controlled time, respectively. Figure 13 shows an example of the simulation results when comparing the vehicle positions with the ground truth. The steering angles are produced by the CNN model trained with data augmentation.

In our experiment, the CNNs trained with and without augmented data are both evaluated using the simulator, and the results are shown in Table I. The error of position is only evaluated when the vehicle is in autonomous driving mode. The data during manual controlled time in simulation are not evaluated. The percentage of autonomous driving time using the model trained with augmented data is 98.32% and number of failures is 9, which are significantly better than the result of 82.09% and 98 without augmented data.

In addition, the simulation results also show that the error of steering wheel angle is not an effective metric for performance evaluation. The model trained with augmented data has mean error of 0.3042 degrees and standard deviation of 1.6029 degrees. The model trained without augmented data has mean error of 0.3118 degrees and standard deviation of 1.2043 degrees. It can hardly tell which model is better from the mean error and standard derivation of steering angles.

The deployed simulator with CNN predictor runs at approximately 13 frames per second (FPS). Considering the input data at 10 Hz, the end-to-end lane keeping system is able to run at real-time. The hardware platform is a desktop computer with Intel i5 3570 K processor running at 3.4 GHz, 32 GB DDR3 RAM and one NVIDIA GTX 1080 GPU.

FIG 12 Example of original image and augmented images given arbitrary vehicle poses. (a) Original image. (b) Augmented image as if the vehicle is shifted right by 0.5 m. (c) Augmented image as if the vehicle is rotated left by 7 degrees. (d) Augmented image as if the vehicle is shifted right by 0.5 m and rotated left by 7 degrees.

V. Discussion

It is worth investigating the causes of some failures during evaluation. 4 out of 9 failures are possibly due to the shadows on the road and unclear lane markings. For example, a failure case is shown in Figure 14. The vehicle is moving out of lane to the right possibly because the front vehicle is changing lane and lane markings are partially blocked. Another case is shown in Figure 15 with casting shadow on the road. Color distortion is observed among the other 5 failure cases. In addition to the limitation of the model, we believe the quality of the input data plays a role in those failures, which can be attributed to factors such as shadows on the road, extreme lighting conditions, camera exposure settings and etc. Because of the complicated scenarios in the real world, the robustness of a model needs be fully examined prior to deployment. Therefore, a simulator built on the real world data becomes very useful.

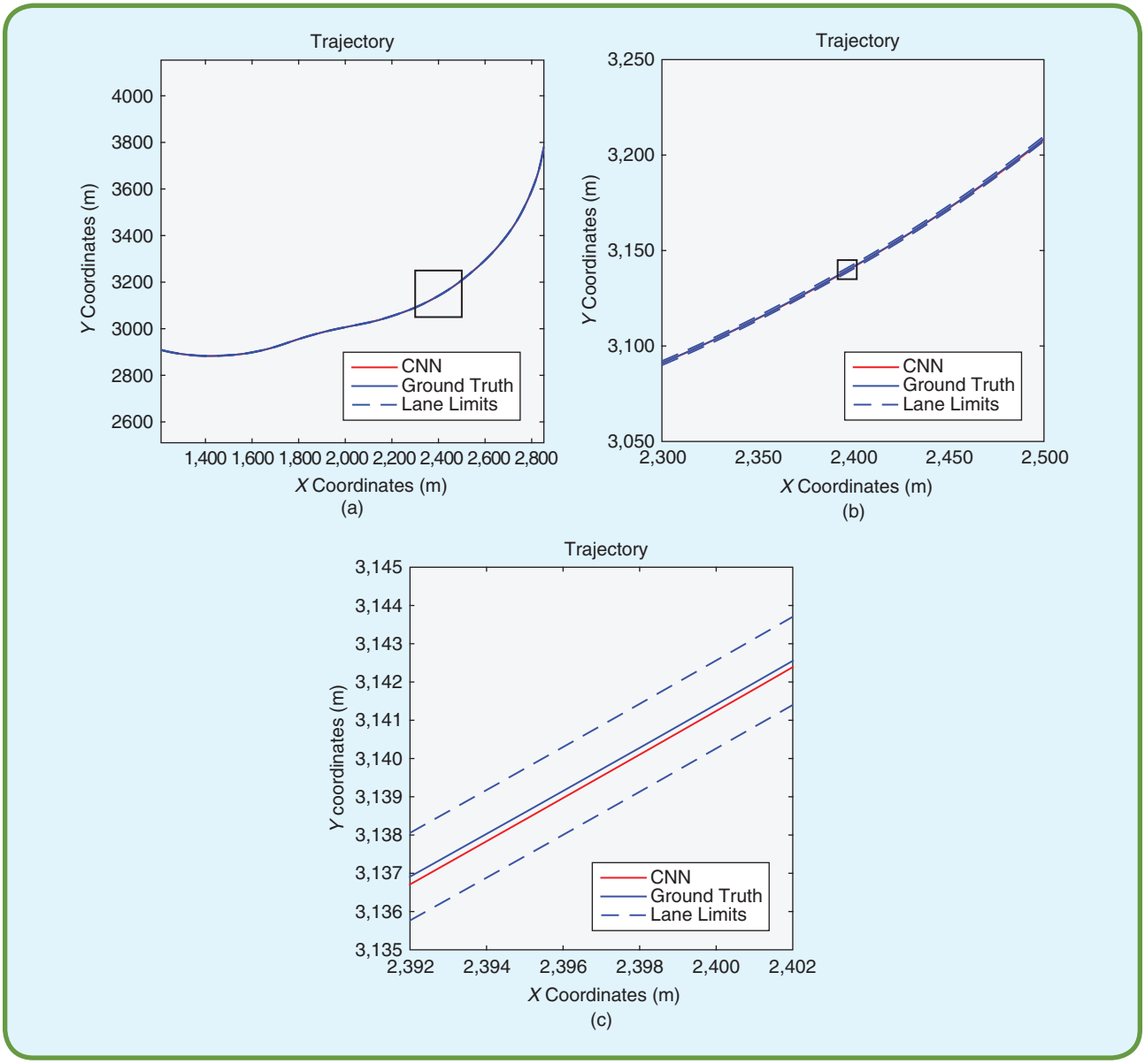


FIG 13 An example of the simulation result, produced by the CNN trained with data augmentation. (a) Overview of the trajectory in a test sequence. (b) Trajectory zoomed-in in the black rectangle in (a). (c) Trajectory zoomed-in in the black rectangle in (b).

Table I. Evaluation result using the simulator, with and without augmented data.

Augmented Data	Autonomy	No. of Failures	Error of Position (Meters)	
			Mean	Standard Deviation
Yes	98.32%	9	0.2179	0.1813
No	82.09%	98	0.2670	0.2071

VI. Conclusions

In this work, we build an autonomous lane keeping simulator based on the recorded data and vehicle dynamics modeling. The simulator can generate a large amount of augmented data for training the CNN model in an end-to-end learning system. Hereby, we propose a state-of-the-art CNN model for autonomous lane keeping, and our experimental evaluations show that training with the simulator results significantly outperforms training with only the recorded data. The failure occurrence is reduced from 98 to 9. The percentage of autonomous driving time is improved from 82.09% to 98.32%. Details of the simulator, including image projection, camera calibration, vehicle dynamics, and trajectory tracking are discussed thoroughly. In addition, the experimental dataset is made available online for autonomous vehicle research and education.

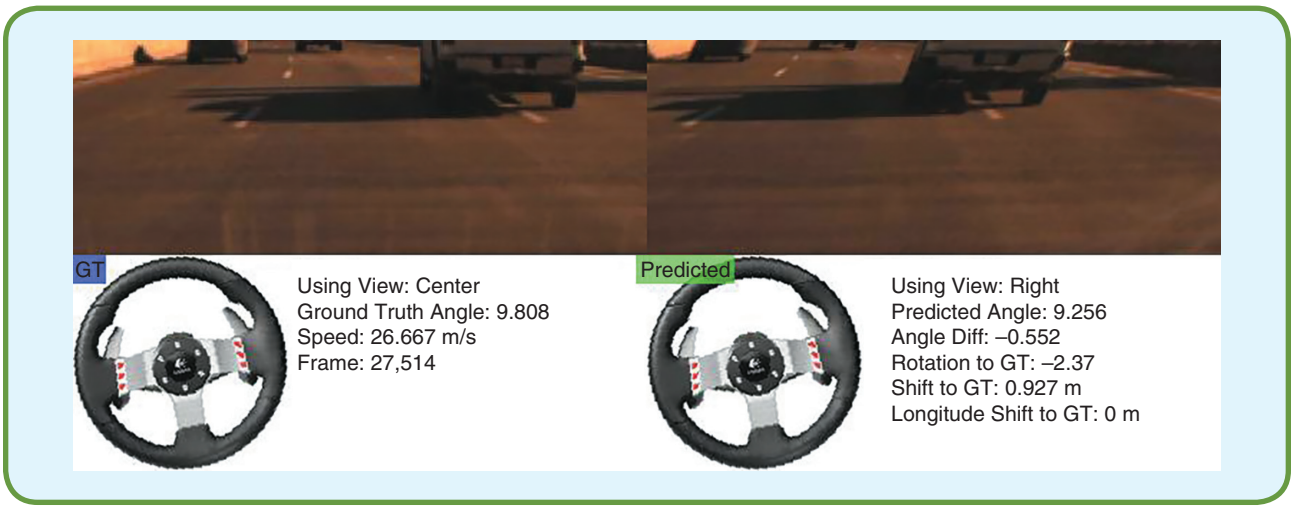


FIG 14 Failure case: the vehicle is moving out of lane to the right because lane markings are partially blocked by the front vehicle.

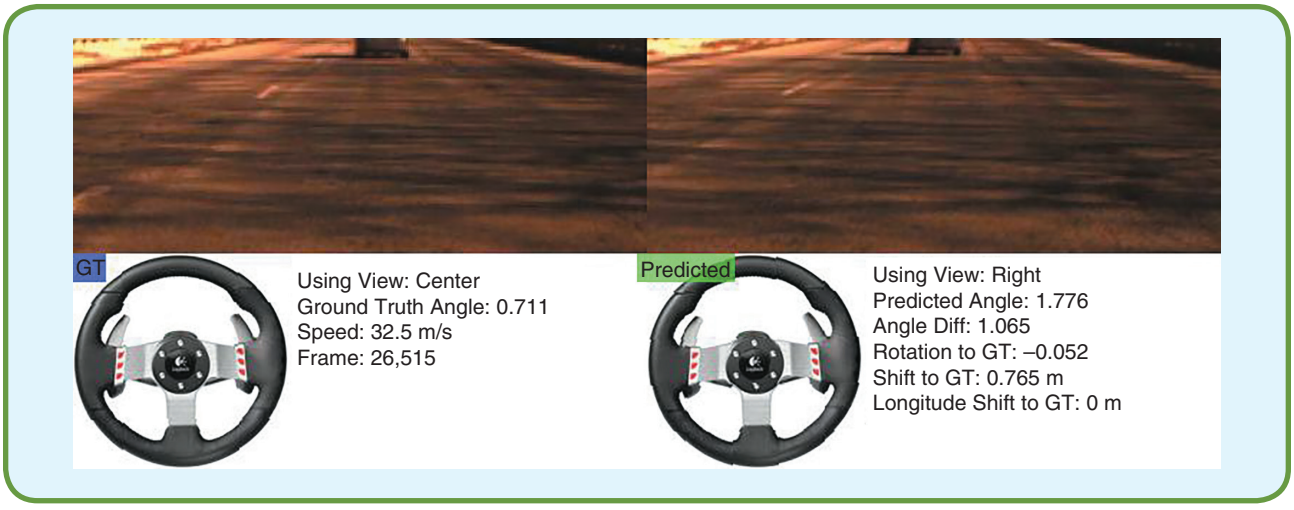


FIG 15 Failure case: the vehicle is moving out of lane to the right because of unclear lane markings.

About the Authors

Zhilu Chen received the B.E. degree in microelectronics from Xi'an Jiaotong University, Xi'an, China, in 2011, and the M.S. degree in electrical and computer engineering from Worcester Polytechnic Institute (WPI), Worcester, MA in 2013. He is currently pursuing the Ph.D. degree from the Electrical and Computer Engineering Department, Worcester Polytechnic Institute, Worcester, MA. His research interests are computer vision, machine learning and GPU acceleration for Advanced Driver Assistance Systems.

Lening Li received the M.S. degree in Computer Science at Worcester Polytechnic Institute (WPI), Worcester, MA in 2016, finished the B.E. degree in Computer Science and B.A. degree in English Language and Literature and graduated with University Honors from Harbin Institute of Technology (HIT), Harbin, China in 2014. He currently is a Ph.D. student in Robotics Engineering at WPI, Worcester, MA. His research interests are in the areas of Optimal Control, Inverse Optimal Control, Motion Planning, and Formal Methods for high-dimensional nonlinear robotic systems.

Xinming Huang (M'01–SM'09) received the Ph.D. degree in electrical engineering from Virginia Tech, Blacksburg, VA in 2001. He is currently a Professor in the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute (WPI), Worcester, MA. Previously he was a Member of Technical Staff with Bell Labs of Lucent Technologies, from 2001 to 2003. His research interests are in the areas of integrated circuits and embedded systems, with emphasis on reconfigurable computing, wireless communications, information security, computer vision, and machine learning.

References

[1] J. Zhao, B. Xie, and X. Huang, "Real-time lane departure and front collision warning system on an FPGA," in *Proc. IEEE High Performance Extreme Computing Conf.*, Sept. 2014, pp. 1–5.

[2] A. J. Humaidi and M. A. Fadhel, "Performance comparison for lane detection and tracking with two different techniques," in *Proc. Al-Sadeq Int. Conf. Multidisciplinary IT and Communication Science and Applications*, May 2016, pp. 1–6.

[3] C. Li, J. Wang, X. Wang, and Y. Zhang, "A model based path planning algorithm for self-driving cars in dynamic environment," in *Proc. Chinese Automation Cong.*, Nov. 2015, pp. 1123–1128.

[4] S. Yoon, S. E. Yoon, U. Lee, and D. H. Shim, "Recursive path planning using reduced states for car-like vehicles on grid maps," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2797–2813, Oct. 2015.

[5] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intelligent Vehicles Symp.*, June 2015, pp. 1094–1099.

[6] D. Wang and F. Qi, "Trajectory planning for a four-wheel-steering vehicle," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2001, vol. 4, pp. 3320–3325.

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316. Accessed on: Nov. 5, 2018.

[8] R. Risack, N. Mohler, and W. Enkelmann, "A video-based lane keeping assistant," in *Proc. IEEE Intelligent Vehicles Symp. (Cat. No.00TH8511)*, 2000, pp. 356–361.

[9] S. Ishida and J. E. Gayko, "Development, evaluation and introduction of a lane keeping assistance system," in *Proc. IEEE Intelligent Vehicles Symp.*, June 2004, pp. 943–944.

[10] J. F. Liu, J. H. Wu, and Y. F. Su, "Development of an interactive lane keeping control system for vehicle," in *Proc. IEEE Vehicle Power and Propulsion Conf.*, Sept. 2007, pp. 702–706.

[11] A. H. Eichelberger and A. T. McCartt, "Toyota drivers' experiences with dynamic radar cruise control, pre-collision system, and lane-keeping assist," *J. Safety Res.*, vol. 56, pp. 67–73, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022437515001061. Accessed on: Nov. 5, 2018.

[12] J. Gao, Q. Wang, and Y. Yuan, "Embedding structured contour and location prior in siamesed fully convolutional networks for road detection," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2017, pp. 219–224.

[13] Q. Wang, J. Gao, and Y. Yuan, "A joint convolutional neural networks and context transfer for street scenes labeling," *IEEE Trans. Intell. Transp. Syst.*, vol. PP, no. 99, pp. 1–14, 2017.

[14] Y. Li, "Deep reinforcement learning: An overview," *CoRR*, vol. abs/1701.07274, 2017. [Online]. Available: http://arxiv.org/abs/1701.07274. Accessed on: Nov. 5, 2018.

[15] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *CoRR*, vol. abs/1612.04340, 2016. [Online]. Available: http://arxiv.org/abs/1612.04340. Accessed on: Nov. 5, 2018.

[16] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *CoRR*, vol. abs/1610.03295, 2016. [Online]. Available: http://arxiv.org/abs/1610.03295. Accessed on: Nov. 5, 2018.

[17] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *CoRR*, vol. abs/1704.02532, 2017. [Online]. Available: http://arxiv.org/abs/1704.02532. Accessed on: Nov. 5, 2018.

[18] S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, "Learning to drive using inverse reinforcement learning and deep q-networks," *CoRR*, vol. abs/1612.03653, 2016. [Online]. Available: http://arxiv.org/abs/1612.03653. Accessed on: Nov. 5, 2018.

[19] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Proc. Advances in Neural Information Processing Systems 1*, San Francisco, CA, USA, 1989, pp. 305–313. [Online]. Available: http://dl.acm.org/citation.cfm?id=89851.89891. Accessed on: Nov. 5, 2018.

[20] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *Proc. 18th Int. Conf. Neural Information Processing Systems*, Cambridge, MA, USA, 2005, pp. 739–746. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976248.297634.

[21] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. D. Jackel, U. Muller, and K. Zieba, "Visualbackprop: Visualizing CNNS for autonomous driving," *CoRR*, vol. abs/1611.05418, 2016. [Online]. Available: http://arxiv.org/abs/1611.05418. Accessed on: Nov. 5, 2018.

[22] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. D. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *CoRR*, vol. abs/1704.07911, 2017. [Online]. Available: http://arxiv.org/abs/1704.07911. Accessed on: Nov. 5, 2018.

[23] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, June 2017.

[24] J. Hardy and M. Campbell, "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles," *IEEE Trans. Robot.*, vol. 29, no. 4, pp. 913–929, 2013.

[25] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *Proc. IEEE American Control Conf.*, 2001, vol. 1, pp. 43–49.

[26] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Robot. Res.*, vol. 28, no. 8, pp. 933–945, 2009.

[27] R. Y. Hindiyeh, "Dynamics and control of drifting in automobiles," Mar. 2013.

[28] E. Galceran, R. M. Eustice, and E. Olson, "Toward integrated motion planning and control using potential fields and torque-based steering actuation for autonomous driving," in *Proc. IEEE Intelligent Vehicle Symp.*, Seoul, Korea, June 2015, pp. 304–309.

[29] R. DeSantis, "Path-tracking for articulated vehicles via exact and Jacobian linearization," *IFAC Proc. Vol.*, vol. 31, no. 3, pp. 159–164, 1998.

[30] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. European Conf. Computer Vision*, 2016, vol. 9906, pp. 102–118.

[31] S. Minhas, A. Hernández-Sabaté, S. Ehsan, K. Díaz-Chito, A. Leonardis, A. M. López, and K. D. McDonald-Maier, *LEE: A Photorealistic Virtual Environment for Assessing Driver-Vehicle Interactions in Self-driving Mode*. Cham: Springer International Publishing, 2016, pp. 894–900.

[32] E. Santana and G. Hotz, "Learning a driving simulator," *CoRR*, vol. abs/1608.01230, 2016. [Online]. Available: http://arxiv.org/abs/1608.01230. Accessed on: Nov. 5, 2018.

[33] R. Szeliski, *Computer Vision: Algorithms and Applications*. New York, NY, USA: Springer Science & Business Media, 2010.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Information Processing Systems 25*, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. Accessed on: Nov. 5, 2018.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556. Accessed on: Nov. 5, 2018.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2015.

[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding, arXiv Preprint, arXiv:1408.5093, 2014.

ITS

Index Terms—autonomous vehicle, lane keeping, end-to-end learning, computer vision, simulator

Xinming Huang
Professor, Electrical and Computer Engineering
Email: xhuang@wpi.edu
Phone: +1-508-831-5771
Fax: +1-508-831-5491
Address:
ECE Department, WPI
100 Institute Road
Worcester, MA 01609-2280