VTGNet: A Vision-Based Trajectory Generation Network for Autonomous Vehicles in Urban Environments

Peide Cai [ID], Yuxiang Sun [ID], Hengli Wang [ID], and Ming Liu [ID], *Senior Member, IEEE*

Abstract—Traditional methods for autonomous driving are implemented with many building blocks from perception, planning and control, making them difficult to generalize to varied scenarios due to complex assumptions and interdependencies. Recently, the end-to-end driving method has emerged, which performs well and generalizes to new environments by directly learning from expert-provided data. However, many existing methods on this topic neglect to check the confidence of the driving actions and the ability to recover from driving mistakes. In this paper, we develop an uncertainty-aware end-to-end trajectory generation method based on imitation learning. It can extract spatiotemporal features from the front-view camera images for scene understanding, and then generate collision-free trajectories several seconds into the future. The experimental results suggest that under various weather and lighting conditions, our network can reliably generate trajectories in different urban environments, such as turning at intersections and slowing down for collision avoidance. Furthermore, closed-loop driving tests suggest that the proposed method achieves better cross-scene/platform driving results than the state-of-the-art (SOTA) end-to-end control method, where our model can recover from off-center and off-orientation errors and capture 80% of dangerous cases with high uncertainty estimations.

Index Terms—End-to-end driving model, uncertainty-aware visual navigation, imitation learning, path planning.

I. INTRODUCTION

FROM a global perspective, approximately 1.3 million people die yearly due to road traffic [1], and nearly 94% of these are related to human driving errors [2]. Autonomous vehicles (AVs) may play an essential role in reducing this number, whilst alleviating traffic congestion, cutting down air pollution and reducing energy consumption in transportation by as much as 90% [3].
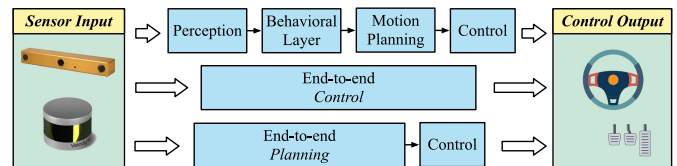
Fig. 1. Different approaches for trajectory planning and decision-making for autonomous vehicles.

In order to achieve autonomous driving, vehicles need to perceive and understand their surroundings [4], then use the extracted information to generate collision-free trajectories to the goal position. However, due to the highly complex environment and an inability to test the system in a wide variety of scenarios [5], achieving universal autonomous driving is still a challenge, especially in real-world urban environments where trajectory generation is a crucial task. Within this framework, the solution trajectory can be represented as a time-parametrized function, $\pi(t) : [0, T] \rightarrow \mathcal{X}$, where T is the planning horizon and $\mathcal{X}$ is the configuration space of the vehicle [6]. Methods for planning and decision-making for AVs can be divided into three main categories [7], [8]: traditional sequential planning, end-to-end control and end-to-end planning, which are illustrated in Fig. 1.

The traditional approaches are usually structured as a pipeline of separate components (e.g., perception, planning and control) linking sensory inputs to actuator outputs [9]. However, there exist some major disadvantages of this approach. 1) Each component needs to be individually specified and tuned, which is difficult to generalize to varied scenarios due to their complex interdependencies [10]. 2) The cost functions in most works have to be carefully designed with complicated parameters. 3) It is prone to error propagation. The accumulated error among unstable modules may lead to false or missed alarms. For example, in the unfortunate Tesla accident in 2016, an error in the perception module, misclassification of a white trailer as sky, propagated down the pipeline until failure [11].

Recently, a new paradigm based on deep learning, which integrates perception, planning and control has achieved impressive results on vehicle navigation [12]–[18]. Called *end-to-end control*, this method formulates the driving problem as learning a mapping from perceptual inputs (e.g., RGB images) directly to vehicle control outputs (e.g., steering angle and throttle). Another similar paradigm is *end-to-end planning* [8], [19]–[21], which maps observations into future trajectories. Deep learning

enables these systems to perform well and generalize to new environments by self-optimising (learning) its behaviour from data, allowing freedom from tuning rule-based parameters in all foreseeable scenarios. However, these works have several shortcomings. 1) The driving ability of recovering from off-center and off-orientation errors has not been considered or validated, which is critical for real-world deployment. 2) Most works only evaluate their models in simple environments with limited complexity in terms of dynamic obstacles and weather/lighting conditions. 3) Additionally, the generalization performance on different types of vehicles with changing physical properties has seldom been evaluated. 4) The neglected uncertainty of the generated driving actions or trajectories prevents the users from checking the correctness of the network output, which is crucial in safety-critical driving tasks. 5) Finally, many works heavily rely on post-processed detailed environmental maps for decision making [20], [21], which are costly to create and transfer to new scenarios.

Towards addressing the above problems, we follow the more generic end-to-end planning paradigm and propose VTGNet, an uncertainty-aware vision-based trajectory generation network for AVs. This paper is an extended version of [8] and the main contributions are summarized as follows.

1) We propose an end-to-end driving network for AVs based on imitation learning, which directly takes as input the raw camera images rather than highly-engineered environmental maps for trajectory generation.
2) We propose a new benchmark *AddNoise* to examine the driving ability to recover from off-center and off-orientation errors.
3) We compare our VTGNet with different baselines and show its superiority under various weather/lighting conditions (e.g., snowy, rainy and foggy) both on a large-scale driving dataset and in a high-fidelity driving simulator with dynamic obstacles.
4) We demonstrate the cross-scene/platform and error recovery driving ability of VTGNet on the *AddNoise* benchmark.
5) We validate the effectiveness of the estimated uncertainty by ablation studies and closed-loop driving tests.
6) We open source our code and collected synthetic driving dataset for future studies on autonomous driving.[1]

II. RELATED WORK

End-to-End Control: ALVINN (autonomous land vehicle in a neural network), proposed by Pomerleau [22] in 1989, is a pioneer attempt to use the neural network for autonomous driving. Due to the composition of a limited number of neural network layers, this method only works in very simple scenarios. In 2016, with the development of convolutional neural networks (CNNs) and computing powers of GPUs, Bojarski *et al.* [12] developed a more advanced driving model named DAVE-2. It achieves autonomous lane following in relatively simple real-world scenarios, such as flat or barrier-free roads, in which the front-view camera is used to stream the images and transmit

[1]https://github.com/caipeide/VTGNet

them into CNNs to compute steering commands. Follow-up works include [16], [13] and [23]. However, these works only target *lane-following* tasks, and since they only consider the camera input for decision making, a wrong turn may be taken at intersections for the lack of high-level navigation commands. Moreover, only simple environments with low-level complexity are considered in these works.

Incorporating Intentions Into Driving Networks: In order to resolve the ambiguities at intersections, driving models proposed by Codevilla *et al.* [15], [18] have been designed to receive not only the perceptual inputs but also high-level driving intentions (i.e., *keep straight* and *turn left*). In this way, the network becomes more controllable. Similarly, Cai *et al.* [24] realize high-speed autonomous drifting in racing scenarios guided by route information with deep reinforcement learning. However, the driving policy is only demonstrated in static enviroments. Hecker *et al.* [25] trained a driving model to use 360-degree camera images equipped with GPS-based route information to predict future steering and speed controls. However, this work was only evaluated *offline* on a pre-collected dataset. Closed-loop *online* driving abilities such as slowing down for pedestrians/vehicles are not presented.

End-to-End Planning Using Temporal Information: To make the driving models more generic, new methods have been recently proposed for end-to-end trajectory planning [8], [19]–[21]. These methods commonly use recurrent neural networks (RNNs) to handle temporal sequential information (e.g., images). For example, Bergqvist *et al.* [19] designed and tested several path planning networks with various types of input including gray-scale images and ego-motions. The results showed that the path generated by long short-term memory (LSTM) or CNN-LSTM is smooth and feasible in many situations. However, this work only considered lane following tasks in simple areas. To handle more complex driving situations, Rhinehart *et al.* [20] consider the future trajectories as a distribution conditioned on post-processed top-view feature maps rather than raw camera images, from which a set of possible paths can be sampled. Similarly, Bansal *et al.* [21] proposed to use top-view environmental representations to generate trajectories with RNNs. However, the detailed top-down views used in these two works are expensive to create, maintain and transfer, and they rely on prebuilt high-resolution maps of the driving areas.

Our work corresponds to the idea of end-to-end planning but differs from the other works in three main aspects: 1) Our method is free from detailed top-view feature/environmental maps and is directly based on highly accessible raw camera images for scene understanding and trajectory generation. 2) The history sequential input is redundant and may contribute differently to the model [26], which, however, is neglected by prior RNN-based driving models. In the spirit of advancements in text translation [27] and graph representation learning [28], we design a *self-attention* LSTM module to better use the temporal information. 3) Our method is extensively evaluated both on a large-scale driving dataset (offline, open-loop) and in a high-fidelity driving simulator (online, closed-loop).

Learning Uncertainty-Aware Navigation Policies: Recently, several methods have been proposed for uncertainty estimation in deep neural networks. Kendall *et al.* [29] proposed that there
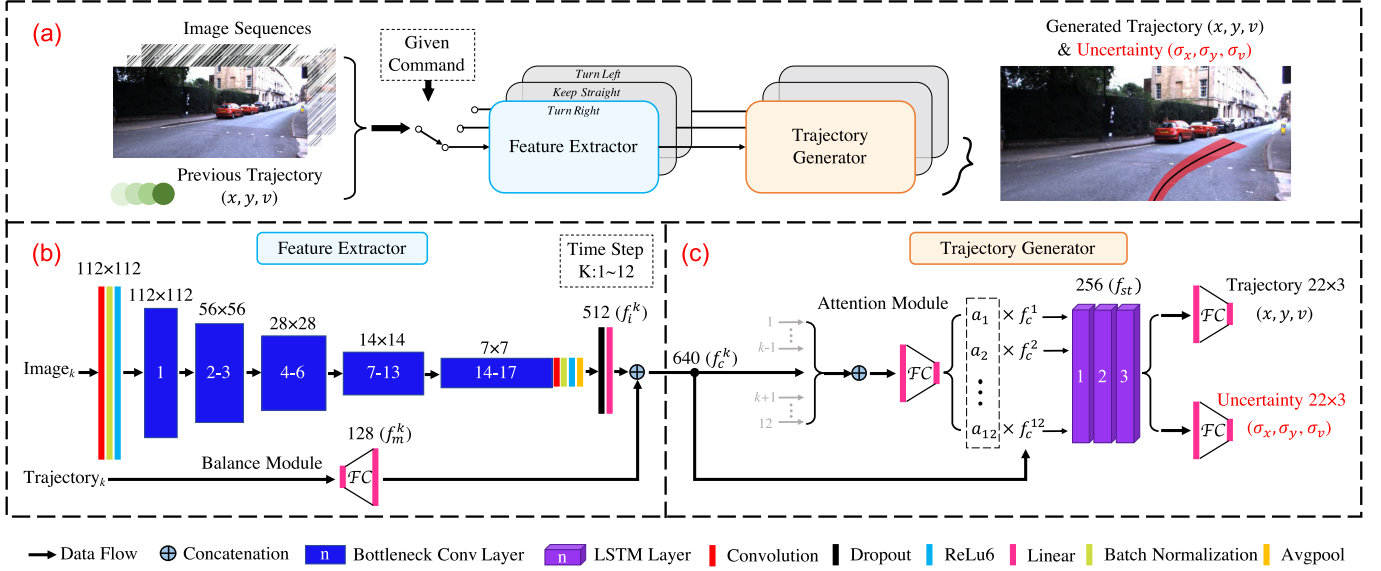
Fig. 2. Architecture of the proposed VTGNet, which consists of a feature extractor and a trajectory generator. VTGNet is composed of three sub-networks that are selectively activated for three common tasks: *turn left*, *turn right* and *keep straight*. MobileNet-V2 is used as the feature extractor with 17 bottleneck convolutional layers. LSTM is used to process the spatiotemporal information. We also adopt a self-attention mechanism to measure the importance of the past sequential information before feeding it into the LSTM. The output of VTGNet is two-branched: 1) A vector of size 22×3 indicating the trajectory in the future 22 frames (velocity and x, y positions in the body frame), and 2) a vector of size 22×3 indicating the uncertainty of the generated trajectory.

are two types of uncertainties: the *aleatoric* uncertainty and the *epistemic* uncertainty. The *epistemic* uncertainty is the model uncertainty, which can be reduced by adding sufficient data. *Aleatoric* uncertainty, also known as data uncertainty, captures the uncertainty from the observed input data. Tai *et al.* [30] accomplished uncertainty-aware visual-based navigation by using data uncertainty to choose the safest action among multiple network outputs. Fan *et al.* [31] also used the data uncertainty to achieve resilient behaviors in prior unknown environments for indoor navigation. Similar to these works, we show performance improvements by explicitly modeling the data uncertainty into deep networks. Differently, we find this method is still insufficient for hard-coded safety on autonomous driving, which opens possible avenues for future research in this area.

III. METHODOLOGY

A. Network Architecture

The overall architecture of the proposed VTGNet is shown in Fig. 2(a). At each time step t, the inputs to the network are camera images $\mathcal{I}_t = \{\mathcal{I}^1, ..., \mathcal{I}^{12}\}$ and the movement information of the vehicle $\boldsymbol{m}_t = \{\boldsymbol{m}^1, ..., \boldsymbol{m}^{12}\}$ in the past 12 frames. The output of the network is the predicted trajectory $\mathcal{T}_t$ and corresponding uncertainty σ_t, which are conditioned on the high-level driving command $\boldsymbol{c}_t$ and observations. The preview horizon of the generated trajectory is set to 3 s (i.e., 22 frames), which is double the average human perception-brake reaction time (RT) of 1.5 s to stop a vehicle [32]. Therefore, the generated trajectory can be denoted as $\mathcal{T}_t = \{\mathcal{T}^1, ..., \mathcal{T}^{22}\}$. Here, $\boldsymbol{m}^k \in \mathbb{R}^3$ and $\mathcal{T}^k \in \mathbb{R}^3$ both contain the velocity and x, y positions in the current body frame. Similarly, the trajectory uncertainty is denoted as $\sigma_t = \{\sigma^1, ..., \sigma^{22}\}$, where $\sigma^k \in \mathbb{R}^3$ contains uncertainties of the velocity and x, y positions.

For implementation, we construct three sub-networks that can be selected by the given command $\boldsymbol{c}_t$ to conduct different tasks, i.e., *turn left*, *turn right* and *keep straight*, similar to other works [15], [18]. In practice, as in [29], we train the network to predict the log variance $\log \sigma^2$ because it is more numerically stable than regressing the variance.

1) The Feature Extractor: Once a specific branch is chosen by a high-level command, the images $\mathcal{I}_t$ are first processed separately and in turn by an image module F_i implemented with CNNs, which extracts a feature vector $f_i \in \mathbb{R}^{512}$ from each image. We use MobileNet-V2 [33] as F_i to extract visual features. It consists of 17 bottleneck convolutional blocks, and each block uses the depthwise separable convolutions rather than a fully convolutional operator. We refer readers to [33] for more details of this module. Then, a balance module F_b implemented with fully connected (FC) layers expands the dimension of each history movement vector $\boldsymbol{m}^k$ from 3 to 128 (f_m^k) to balance the influence of the vision and motion feature vectors, for which we draw on the experience of [34]. The outputs of these two modules are then concatenated together at every history time step into combined vectors of length 640, represented by

$$f_c^k = < F_i(\mathcal{I}^k),\ F_b(\boldsymbol{m}^k) >,\ 1 \le k \le 12, \qquad (1)$$

where $< \cdot >$ represents the concatenation operation, $F_i(\mathcal{I}^k)$ is the output of the image module for the k-th image, $F_b(\boldsymbol{m}^k)$ is the output of the balance module for the k-th motion state vector, and f_c^k is the k-th combined vector.

2) The Trajectory Generator: To better use the sequential information, the set of $f_c = \{f_c^1, ..., f_c^{12}\}$ is first concatenated and processed by FC layers to generate attentions $a_1, ..., a_{12}$ to measure the relative importance of the past information. We use *softmax* to activate the last layer so that $\sum_1^{12} a_k = 1$. Then, we feed the modulated features $\{a_1 f_c^1, ..., a_{12} f_c^{12}\}$ into a
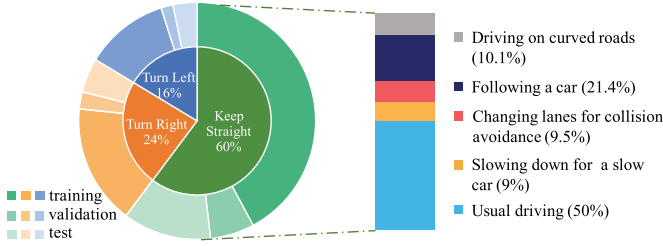
Fig. 3. Dataset distribution and splitting. For each situation, the split ratio of training, validation and test is 7:1:2. The staked column bar on the right shows the distribution of data in the *keeping straight* situation, where we have balanced the portion for usual and unusual driving cases. The turning situation does not contain significant interaction with the road agents.

three-layer LSTM to generate the spatiotemporal feature $f_{st} \in \mathbb{R}^{256}$ of the surroundings. It is further compressed by 2 FC layers to get two vectors, both of size 22×3, which represent the trajectory $\mathcal{T}_t$ and uncertainty σ_t generated at time step t for the future 22 frames.

3) Loss Function: As the name suggests, the basic idea of imitation learning is to train a network that mimics human behaviors, which can be solved with supervised learning. Let $\mathcal{T}_t'$ denote the expert-provided trajectory in the training dataset at sample time t, and θ denote the learnable parameters of the network F. Then, the optimal parameters θ^* can be trained by minimizing the average prediction loss:

$$\theta^* = \arg \min_{\theta} \sum_t \mathcal{L}\left(F\left(\mathcal{I}_t, \boldsymbol{m}_t, \boldsymbol{c}_t; \theta\right), \mathcal{T}_t'\right), \qquad (2)$$

where $\mathcal{L}$ is the per-sample loss at sample t. We follow the training method for data uncertainty in [29] and define $\mathcal{L}$ as

$$\mathcal{L} = \frac{\|\mathcal{T}_t - \mathcal{T}_t'\|_2^2}{2\sigma_t^2} + \frac{1}{2}\log\sigma_t^2. \qquad (3)$$

B. Dataset Generation and Analysis

We build our dataset from the RobotCar dataset, which was recorded in dynamic urban areas from May 6, 2014 to November 13, 2015. This dataset captures scenarios with various weather and lighting conditions, along with some long-term changes such as construction and roadworks. The front-view images are recorded by a Point Grey Bumblebee XB3 camera on top of the vehicle. In addition, the ground-truth positions and velocities are acquired from the fused GPS/inertial solution at a frequency of 50 Hz.

1) Data Generation: From the RobotCar dataset, we aim to extract both camera images and movement information to train and test VTGNet. Therefore, the data without GPS information is first *filtered* out and a total of 29 driving routes are collected. Another issue that needs to be considered is *data balance*. Specificallly, we need to balance the portion for different driving cases (e.g., car following, slowing down for collision avoidance, etc.) and the distribution of driving samples in varied environmental conditions. The final dataset distribution is shown in Fig. 3 and in Table I.

This data is then *reconstructed* for training and testing in this work, where we need to equip every image with the trajectory of the ego-vehicle in the past 1.5 s and the future 3 s. For

TABLE I
THE ENVIRONMENTAL DISTRIBUTIONS OF OUR DATASET

Environments	# Turn Left	# Turn Right	# Keep Straight
Sun	3705	5355	7921
Rain	3369	4401	16548
Snow	1280	3975	6089
Dusk	1059	1194	4836
Night	2339	2705	10930
Overcast	2733	3103	7016

Fig. 4. Dataset distribution map. We plot and cluster the locations of all the samples from our dataset. Clusters with more samples have a brighter color and are larger in size.

implementation, we first interpolate the ground-truth universal transverse mercator (UTM) position and velocity series to the image timestamps, which are recorded at 15 Hz, and then convert their coordinates to the vehicle body frame. After this, we manually label every image with a corresponding intention command indicating the driving direction of the vehicle based on the ground-truth trajectories. For example, when the ego-vehicle approaches an intersection and plans to take a right turn, we change the command from *keep straight* to *turn right*. When the steering is completed, we recover the command to *keep straight*. Finally, we crop the raw images to the shape of (1247, 384) by removing the sky and hood on the top/bottom of the images because these areas are less informative for scene understanding. We believe that the learning process is more efficient by doing so.

2) Data Analysis: Our final dataset contains 88,558 images of driving sequences in Oxford for 61.52 km, which covers 6 environmental conditions: *overcast, sun, rain, snow, dusk* and *night*. Each environment has a special style of visual appearance. For example, in the *rain* scenario, the roads are often covered by fallen leaves, and sometimes raindrops on the camera lens cause blurry areas in the image. In the *night* scenario, the shutter speed is much slower than it is in the daytime, which can lead to motion artifacts around objects when the vehicle moves. In the *snow* scenario, the undrivable areas are often covered by white snow, contrasting sharply with the drivable areas. Sample visual appearances can be seen in Fig. 8. In addition, to visualize how the dataset distributes geologically, we plot the locations on an aerial map in Fig. 4.

TABLE II
MODELS: THE QUANTITATIVE EVALUATION RESULTS FOR DIFFERENT MODELS. *ENVIRONMENTS*: THE EVALUATION RESULTS OF VTGNET IN DIFFERENT WEATHER/LIGHTING CONDITIONS. SMALLER NUMBERS ARE BETTER. THE BOLD FONT HIGHLIGHTS THE BEST RESULT IN EACH COLUMN

| Models	Accel (m/s^2)	$\mathcal{E}_v$ (m/s)	$\mathcal{E}_{acc}$ (m/s^2)	$\mathcal{E}_{ad}$ (m)	$\mathcal{E}_x$ (m)	$\mathcal{E}_y$ (m)	$\mathcal{E}_{fd}$ (m)
CNN-FC [16]	**0.181**	1.104	0.313	1.621	0.268	1.545	3.271
CNN-LSTM [16]	0.488	0.995	0.564	1.461	0.269	1.380	2.907
CNNState-FC [19]	0.326	0.477	0.399	0.628	0.242	0.521	1.444
VTGNet *(ours)*	0.325	**0.289**	**0.311**	**0.426**	**0.188**	**0.335**	**1.036**
Environments							
Sun	0.402	0.286	0.339	0.470	0.193	0.372	1.120
Rain	0.294	0.270	0.299	0.409	0.172	0.329	0.962
Dusk	0.350	0.359	0.353	0.421	0.185	0.332	1.095
Night	0.318	0.310	0.308	0.414	0.205	0.310	1.040
Overcast	0.337	0.311	0.336	0.508	0.204	0.418	1.226
Snow	**0.262**	**0.236**	**0.247**	**0.329**	**0.171**	**0.237**	**0.818**
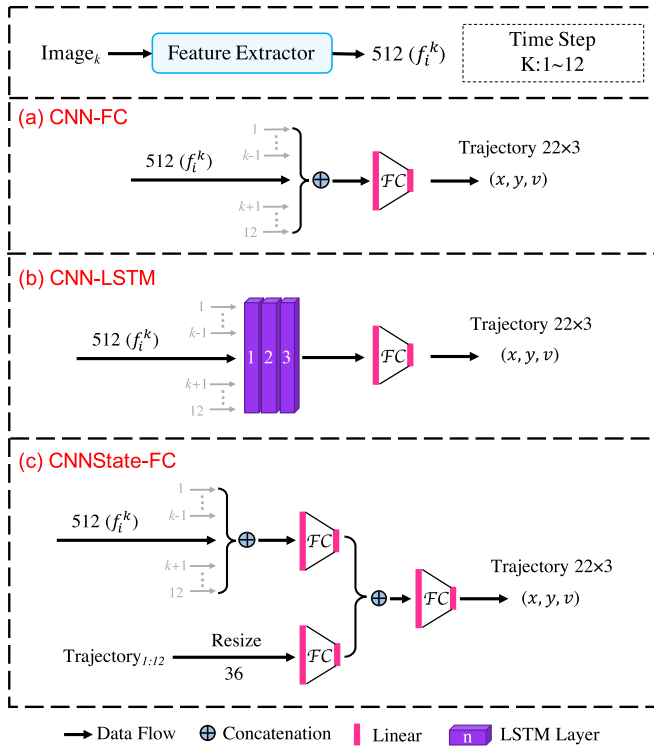
Fig. 5. Different baselines for comparison. The feature extractor for these networks is the same as that in our proposed VTGNet shown in Fig. 2.

IV. EXPERIMENTS ON THE ROBOTCAR DATASET

A. Training Details and Baselines

We implement the proposed VTGNet with Pytorch, and train it on the dataset introduced in Section IV with the NVIDIA 1080Ti graphics card. The split ratio of training, validation and test is set to 7:1:2. We use the Adam optimizer[35] with an initial learning rate of 0.0001 and batch size of 15. The network is trained to converge when no further decrease in the validation loss can be observed. For comparison, we also train and fine-tune three other vision-based end-to-end trajectory

planning baselines on the same training set. They are introduced as follows, and are shown in Fig. 5.

CNN-FC. This network takes as input the image sequences. The CNNs first extract the visual features in the past 12 frames (1.5 s), and then these features are concatenated together to be compressed by FC layers. This method follows the idea of *TCNN* introduced in [16].

CNN-LSTM. This network takes as input the image sequences. The extracted features in the past 12 frames are processed by a three-layer LSTM block into a vector $\in \mathbb{R}^{512}$. The vector is then compressed by FC layers. This method follows the idea of *CNN-LSTM* introduced in [16].

CNNState-FC. This network takes as input both image and movement sequences. The extracted features are concatenated and compressed with FC layers. This method follows the idea of *Merging Model* introduced in [19].

We adopt 7 metrics to evaluate the performance of different networks. These are the average values computed over the entire test dataset.

- *Accel* measures the smoothness of the generated trajectory, which is the average acceleration of the preview time. The lower its values, the smoother and more comfortable the corresponding trajectories.
- $\mathcal{E}_v$ and $\mathcal{E}_{acc}$ measures the mean velocity and acceleration error, respectively, in the preview horizon of a trajectory.
- $\mathcal{E}_{ad}$ represents the average displacement error [36]. It is the L2 distance between the generated and the ground-truth trajectory.
- $\mathcal{E}_x$ and $\mathcal{E}_y$ represents the lateral and longitudinal error, respectively, of a trajectory.
- $\mathcal{E}_{fd}$ is the final displacement error [36], which means the L2 distance between the final waypoints of the generated and ground-truth trajectories.

B. Quantitative Analysis

1) Comparison to Baselines: We first show the evaluation results on the test set from different model architectures in Table II-*Models*.
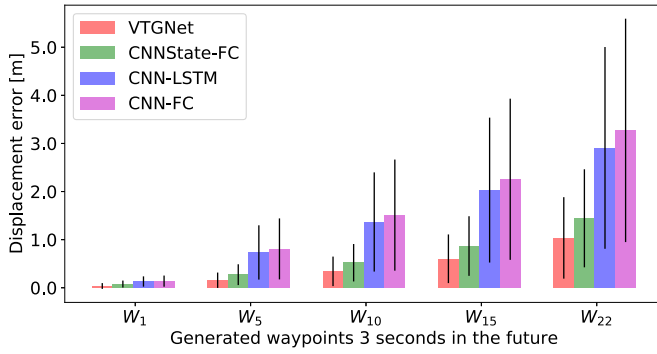
Fig. 6. Displacement error on the generated waypoints for different models. The black lines indicate the standard deviation.

Trajectory Smoothness: Table II reveals that `CNN-FC` generates the smoothest trajectories with the lowest *Accel* value (0.181). `VTGNet` ranks second place (0.325). `CNN-LSTM` performs worst on this metric with the highest *Accel* (0.488).

Error Analyses: From the second to the last column of Table II, we can see that `VTGNet` generates more accurate trajectories than the other models in terms of the error-related metrics such as $\mathcal{E}_v$ and $\mathcal{E}_{ad}$. This superior performance indicates that our model performs more like humans than the other models. The displacement error over the 3.0 s preview horizon from different models is shown in Fig. 6. Their errors and related variance all increase as the preview time increases, and our `VTGNet` achieves the best results with the smallest values. In addition, we can infer the reasonability of our network design by comparing the results from different baseline models, which are discussed as follows.

Table II shows that both `CNN-LSTM` and `CNNState-FC` perform better than `CNN-FC` with less errors. The superiority of `CNN-LSTM` indicates that the recurrent architecture, such as LSTM, is much more efficient than the FC layers in processing spatiotemporal information. We conjecture that the LSTM module can infer the hidden state, such as environmental changes among the frames, which leads to better performance on this imitation learning task. However, it is difficult for the FC layers to extract valuable information directly from the consecutive visual features. On the other hand, the superiority of `CNNState-FC` indicates that past movement is beneficial for trajectory generation, which functions as the *state memory*, similar to the memory of humans. Although the position and orientation information is contained in the raw images, using a CNN alone is not sufficient to extract this information and generate future trajectories.

2) Comparison Among Different Environments: In this section, we test and evaluate our `VTGNet` separately on different environments with various lighting and weather conditions, with the aim to see whether this architecture can generalize to different visual appearances. The results are shown in Table II-*Environments*. It can be seen that `VTGNet` achieves robust results under different conditions and performs best in the *snow* environment. In this scene, the generated trajectories have the lowest acceleration (0.262 m/s^2) and error results ($\mathcal{E}_v, \mathcal{E}_{ad}$, etc.). This is probably because the drivable areas are more distinct in this scenario with the undrivable areas covered by white snow. Another possible reason for the lower *Accel* value

TABLE III
THE DRIVING STYLE REPRESENTED AS THE MEAN AND STANDARD DEVIATION OF THE DRIVING ACCELERATION (m/s^2) FOR THE TRAINING DATASET AND VTGNET

Environments	Training Data	VTGNet
Sun	0.38 ± 0.34	0.40 ± 0.35
Rain	0.39 ± 0.35	0.29 ± 0.24
Dusk	0.34 ± 0.31	0.35 ± 0.32
Night	0.32 ± 0.34	0.32 ± 0.30
Overcast	0.32 ± 0.33	0.34 ± 0.29
Snow	0.21 ± 0.25	0.26 ± 0.22

Fig. 7. Sample results from different baselines and our `VTGNet`. The black lines indicate the planned results and the green lines indicate the ground truth. The red shaded areas represent the estimated uncertainty.

on snowy days is that the reference training data from humans reflects that they tend to drive with less dynamics on slippery roads in such weather, which affects our `VTGNet` trained with imitation learning. We verify this idea by quantitatively analyzing the driving styles of the training trajectories from humans and the generated trajectories from `VTGNet`. We formulate the driving style as the mean and standard deviation of the driving accelerations, and the results are shown in Table III. It can be seen that in most conditions the driving styles of humans and `VTGNet` are quite similar, which accords with the principle of imitation learning.

C. Qualitative Analysis

1) Comparison to Baselines: We compare `VTGNet` with the baselines, and the results are shown in Fig. 7. It can be seen that although `CNN-FC` generates smooth trajectories, it deviates significantly from the ground truth values when taking turns in Fig. 7(a). In Fig. 7(b), the ego-vehicle should slow down for a pedestrian ahead, while `CNN-FC` maintains the current speed and moves forward. `CNN-LSTM` performs better in terms of errors, but the planned results from this model are rather jerky. In general, `VTGNet` achieves the best results.

2) Results in Different Environments: Fig. 8 displays sample qualitative results of `VTGNet` in 5 typical environments, in which the challenging scene *rainynight* is presented in the last row. Because of the raindrops on the camera lens, the halo effects in this scene are sometimes conspicuous. It is worth
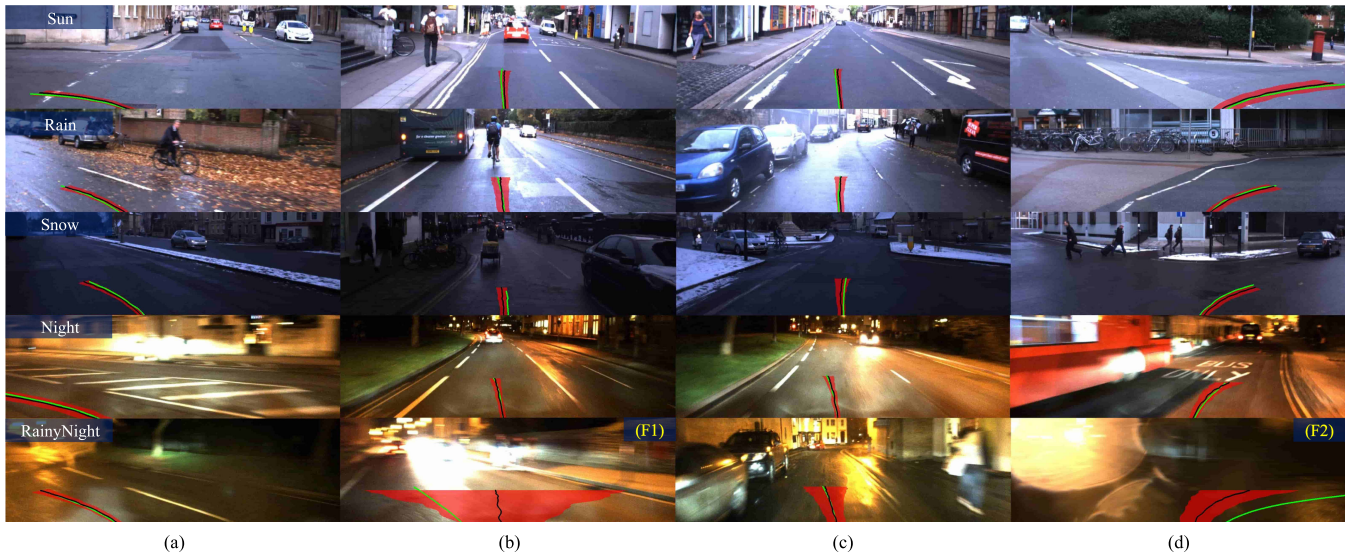
Fig. 8. Sample qualitative results for our model VTGNet in various environments. Column (a) shows the results for *turn left*, and column (d) for *turn right*. Column (b) shows the results for car following, and column (c) for lane keeping. The black lines indicate the planned results and the green lines indicate the ground truth. The red shaded areas represent the estimated uncertainty.

Fig. 9. Error values of ablated models ($\mathcal{M}_1 - \mathcal{M}_3$) and our VTGNet ($\mathcal{M}_0$). Lower values are better.

Fig. 10. Sample results from ablated models ($\mathcal{M}_1 - \mathcal{M}_3$) and our VTGNet ($\mathcal{M}_0$). The black lines indicate the planned results and the green lines indicate the ground truth. The red shaded areas represent the estimated uncertainty.
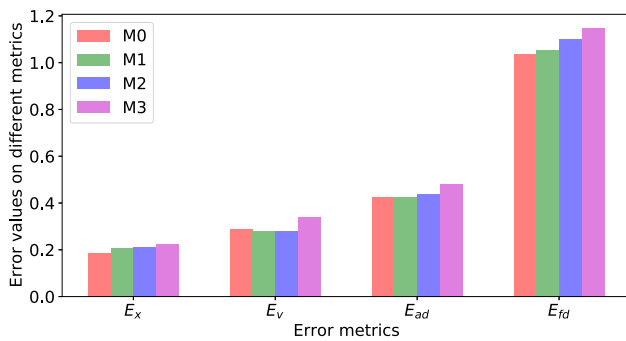
noting that this scene is not included in the training set, and is only used to qualitatively test the generalization performance. In general, we can see that our VTGNet is able to generate collision-free trajectories under various lighting and weather conditions. The performance is also reliable in some tough scenarios. For example, in *night*-(d), the bus in the left area presents a severe motion artifact, and VTGNet generates a safe trajectory to turn right. In *rainynight*-(c), the motion artifact is more severe but the generated results are still reasonable and similar to the ground truth (with higher uncertainty). We consider that the well-balanced training dataset of high diversity in terms of lighting and weather conditions contributes to such generalizable performance.

Additionally, we show some failure cases, indicated by extreme uncertainties in *rainynight*-(b,d), where the input image is heavily noised. In such scenarios, the system should drive the vehicle more cautiously to avoid potential accidents.

D. Ablation Studies

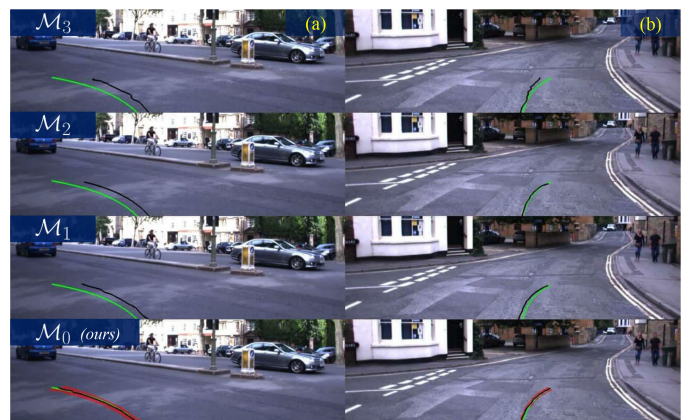We train and test a series of ablated models $\mathcal{M}_1 \sim \mathcal{M}_3$ to show the benefits of the VTGNet design ($\mathcal{M}_0$). $\mathcal{M}_1$ removes the uncertainty estimation from $\mathcal{M}_0$. $\mathcal{M}_2$ further removes the self-attention module from $\mathcal{M}_1$. $\mathcal{M}_3$ is a variant of $\mathcal{M}_2$, which adopts two LSTM modules to process image and movement sequences, separately. We show the error values from these models in Fig. 9. In general, the performance of the model decreases from $\mathcal{M}_0$ to $\mathcal{M}_3$, indicating the rationality of our VTGNet. The superiority of $\mathcal{M}_0$ over $\mathcal{M}_1$ is consistent with the finding in [29] where modeling data uncertainty improves prediction performance. On the other hand, $\mathcal{M}_3$ performs worse than $\mathcal{M}_2$, from which we can conclude that using a single LSTM module to process the *multi-modal* information together would help to generate more reasonable outputs. Furthermore, we show a qualitative comparison in Fig. 10, where the results from $\mathcal{M}_0$ are both accurate and smooth.
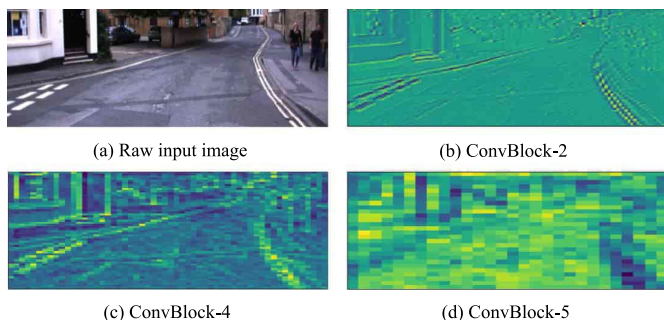
Fig. 11. Image feature visualization of our `VTGNet`. (a) The input RGB image. (b–d) The averaged feature maps of some bottleneck convolutional layers in the feature extractor.

Fig. 12. Attention distribution from our `VTGNet` in two typical scenarios. (a) Turning at intersections. (b) Lane following.

E. Discussion and Reasoning

1) Feature Extractor Analysis: Compared with hand-crafted feature extractors, one advantage of CNNs is that they can learn to extract useful features automatically from raw RGB images. We confirm this idea by visualizing the averaged feature maps from a series of convolution layers of our model. The results are shown in Fig. 11. We can see that the obvious extracted features are the road boundaries, which are helpful for generating the trajectories.

2) Attention Analysis: Here, we analyze the attention mechanism of our model using Fig. 12. We can see that when taking turns (Fig. 12(a)), the observations at intersection entrances (time step 3) are weighted with more attentions, which contains more structural information of the surroundings. On the other hand, when following lanes on roads, more recent observations are assigned with more attention because the past observations are rather redundant, providing no extra valuable information.

V. CLOSED-LOOP EXPERIMENTS IN THE CARLA DRIVING SIMULATOR

In this section, we aim to determine whether our `VTGNet` can perform more complicated tasks, such as recovering from mistakes and driving in dynamic traffic. To this end, we conduct closed-loop experiments in the CARLA driving simulator [37] to further show its performance. CARLA is an open-source simulator providing a high-fidelity dynamic world and different vehicles with realistic physics.

A. Implementation Details

1) Domain Adaption to New Environments: Our prior model is trained with the Robotcar dataset, which lacks reference trajectories to recover from off-center or off-orientation mistakes. On the other hand, the domain gap in terms of visual difference also prevents us from directly deploying the model in CARLA. To improve the performance, we first collect a new expert driving dataset in CARLA, named VTG-Driving, and then refine our model with this dataset to achieve *domain adaption*. During data collection, we set random routes ranging from 300 m to 1500 m in Town01 of CARLA and drive the vehicle at the desired speed of 40 km/h. The direction command is given by a high-level global planner of CARLA. To construct a dynamic environment, we set five weather conditions: *clear-day*, *clear-sunset*, *foggy-day*, *rainy-day*, and *rainy-sunset*. For each weather condition, we run 100 episodes with dynamic roaming pedestrians and vehicles that are controlled by the AI engine of CARLA. Furthermore, we add random *steering* noise to the vehicle every 6 s to collect error-recovery trajectories. The final dataset lasts 16.6 hours and covers a driving distance of 288.7 km, which is also released for public research.

2) Evaluation Benchmark: The prior benchmarks of CARLA mainly concentrate on the ability to handle complex visual appearances [37] or the ability to drive in urban traffic with dynamic obstacles [18]. We extend these ideas and propose a much tougher benchmark *AddNoise* to also evaluate the ability to recover from off-center/orientation errors. Similar to the data collection process, we add random *steering* noise to the test vehicle every 5 s that lasts 0.2∼1.0 s. Moreover, we examine the cross-platform ability by changing the training *car* to an unseen *motorcycle*, which is smaller in size and has a smaller turning radius and larger acceleration.

In this benchmark, the test vehicle should drive in empty and dynamic (with other dynamic vehicles/pedestrians) environments against the noise introduced above. Each task corresponds to 60 goal-directed episodes under the 5 weather conditions introduced in Section V-A1. In each episode, the vehicle starts from a random position and is directed by a high-level planner to reach the destination. The episode is considered to be successful if the vehicle arrives at the destination within the time limit and does not collide with other objects. Finally, the success rate is used to measure the autonomous driving ability.

3) Baselines for Comparison: We have shown the superiority of `VTGNet` in the area of end-to-end trajectory *planning* in Section IV. In this section, we further compare it with a recent SOTA vision-based end-to-end *control* network named `CILRS` [18]. This model is trained with data unaffected by noise (filtered from VTG-Driving) following the setup as detailed in the original paper. We also train another model named `CILRS++` on our full dataset including the noise-recovery behaviors. The reference expert actions for training are the original actions that are not

TABLE IV
CLOSED-LOOP EVALUATION RESULTS ON SUCCESS RATE (%) IN THE CARLA DRIVING SIMULATOR

Traffic	Training Conditions		New Vehicle		New Vehicle & Town	
	Empty	Dynamic	Empty	Dynamic	Empty	Dynamic
CILRS [18]	1.7	3.3	8.3	3.3	10.0	3.3
CILRS++	**95.0**	**86.7**	15.0	13.3	23.3	25.0
VTGNet	93.3	75.0	**43.3**	**40.0**	**35.0**	**33.3**

Fig. 13. Closed-loop evaluation results of VTGNet in our *AddNoise* benchmark. Row-1 shows some normal driving behaviors: (a,b) turning at intersections, (c) vehicle-following and (d) stopping for pedestrians. Row-2 shows recovery behaviors against off-center/orientation errors under different weather conditions. The black lines indicate the planned results and the red shaded areas represent the estimated uncertainty.

overlaid with noise. Note that we replace the original image processing backbone of these two models to MobileNet-V2 as our VTGNet for fair comparison. For VTGNet, we design two PID controllers for the test vehicles (car/motorcycle) to translate the generated trajectory into driving actions.

B. Results

1) Quantitative Analysis: The evaluation results are shown in Table IV. Although CILRS achieves good performance in [18], with a reported success rate of 42%∼97% in training conditions, its performance degrades significantly in *AddNoise* and the success rate is only 1.7%∼3.3%. After being augmented with training data that recovers from mistakes, CILRS++ performs much better against the noisy test environment and achieves higher success rate of 86.7%∼95.0% in training conditions. However, these two models performs badly on new platforms, i.e., a motorcycle in this work, and the highest success rate is only 25.0%. The reason is that the learned driving policies of end-to-end control systems can only perform well on data collected with specifically calibrated actuation setups, which also accords with the statements in [16]. Actually, we observe a very shaky and unstable driving performance of these two models where the

motorcycle often rushes to the roadside and collides with static objects such as lamps and fences. Note that the new test town (Town02) in CARLA contains less static objects on the roadside than Town01, and therefore the results for *New Vehicle&Town* are better than for *New Vehicle* for CILRS++.

On the other hand, our VTGNet performs a little worse than CILRS++ in training conditions (75.0%∼93.3%) but achieves much better generalization results on new vehicle and new town. For example, the success rate of our model in dynamic environments of the *New Vehicle* setup is 40.0%, which is three times higher than that of CILRS++. We accredit such superiority to our model architecture, which allows the control module to flexibly adapt to different vehicle platforms. Since the motorcycle has smaller turning angles, it is more susceptible to the *steering* noises than the *car* is for *Training Conditions* in our benchmark. Because of this, very large off-center and off-orientation errors may occur after the noise process. For example, the motorcycle is often disturbed onto sidewalks where the road is out of sight, which is hard to recover and explains the performance degradation from *Training Conditions* to *New Vehicle*.

2) Qualitative Analysis: We show some driving snapshots of VTGNet in Fig. 13. It shows that VTGNet not only adapts
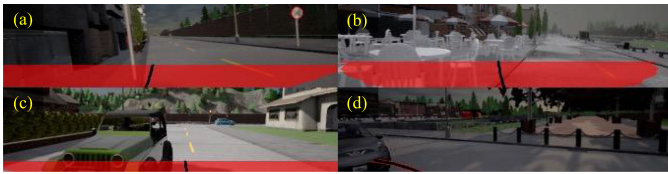
Fig. 14. Failure cases of our driving model. (a–c) Most failure cases (80%) are informed with a high uncertainty output, which can be avoided with timely human intervention. (d) A few failure cases can not be captured with the estimated uncertainty. The black lines indicate the planned results and the red shaded areas represent the estimated uncertainty.

to different weather conditions but also possesses the ability to perform reactive driving such as following a vehicle (Fig. 13(c)) and stopping for pedestrians (Fig. 13(d)). Furthermore, we show its recovery behaviors in Fig. 13(b), where the ego-car is autonomously recovering from off-center/orientation errors disturbed by periodic steering noises. More related results are shown in the supplemental videos[2].

3) Benefits and Limitations of Uncertainty Estimation for Safe Driving: We observe that 80% of failure cases in the benchmark results of `VTGNet` are informed with a high uncertainty output, and thus can be avoided by timely human intervention, as shown in Fig. 14(a–c). However, robust functional safety measures are still beyond reach because in a few cases, the model is too confident in its wrong prediction with low uncertainty, as shown in Fig. 14(d), where the ego-vehicle collides with the vehicle ahead without slowing down. Such limitations leave us possible avenues for future research on safer autonomous driving.

VI. CONCLUSION

In this paper, to improve the driving ability of current learning-based methods in terms of safety measures and recovery behaviors against errors, we proposed an uncertainty-aware vision-based trajectory generation network for AVs, named VTGNet. For acceptable training and testing performance, we first created a large-scale driving dataset from the original Robotcar dataset. Then, we implemented the proposed network with a twofold pipeline. The first part is a feature extractor composed of bottleneck CNN layers based on MobileNet-V2. The second part is a trajectory generator, which consists of a self-attention module to weight the redundant history information, and an LSTM module to process the spatiotemporal features. The overall framework was designed to take as input the front-view image and movement sequences in the past 1.5 seconds with the intention command to generate feasible trajectories 3 seconds in the future. The whole network is differentiable and was then trained end-to-end using imitation learning. Third, we validated the robustness and reliability of our proposed network under various weather and lighting conditions by comparing it with 3 baselines and several ablated models. Finally, we conducted extensive closed-loop experiments in the CARLA driving simulator to demonstrate the more advanced ability of VTGNet to recover from off-center/ orientation errors and cross-platform driving. Furthermore, we showed that most of the failure cases can be avoided by timely

human-intervention with the estimated uncertainty, which is beneficial to develop safety-critical driving systems.

Despite the success of the proposed network, there still exist some limitations of our model. 1) Robust safety measures can not be guaranteed even with the uncertainty estimation. More studies are needed in this area to develop safer autonomous driving systems. 2) The discrete high-level driving commands are not suitable for more complex road topologies such as intersections with more than one possibility to turn. It is expected to incorporate global route information to provide more detailed driving intentions into the network. 3) Camera images may not be reliable under extreme lighting conditions, as shown in Fig. 8-(F1,F2). We cannot expect the system to generate acceptable trajectories with such input data. Therefore, in the future, we aim to develop a framework based on our current work to use multi-modal information from complementary sensors such as lidar, radar and a thermal camera. We believe that by doing so, the system could be more robust in challenging environments.

REFERENCES

[1] W. H. Organization *et al.*, *Global Status Report on Alcohol and Health-2014*. World Health Organization, 2014.

[2] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," NHTSA's National Center for Statistics and Analysis, Tech. Rep. DOT HS 812115, 2015.

[3] A. Brown, B. Repac, and J. Gonder, "Autonomous vehicles have a wide range of possible energy impacts," NREL, University of Maryland, Tech. Rep. NREL/PO-6A20-59210, 2013.

[4] R. Fan, H. Wang, P. Cai, and M. Liu, "SNE-RoadSeg: Incorporating surface normal information into semantic segmentation for accurate freespace detection," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 340–356.

[5] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–22, 2020.

[6] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[7] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control, Robot., Auton. Syst.*, pp. 187–210, 2018.

[8] P. Cai, Y. Sun, Y. Chen, and M. Liu, "Vision-based trajectory planning via imitation learning for autonomous vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2019, pp. 2736–2742.

[9] R. McAllister *et al.*, "Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 4745–4753.

[10] A. Kendall *et al.*, "Learning to drive in a day," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 8248–8254.

[11] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[12] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, arXiv:1604.07316.

[13] J. Jhung, I. Bae, J. Moon, T. Kim, J. Kim, and S. Kim, "End-to-end steering controller with CNN-based closed-loop feedback for autonomous vehicles," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 617–622.

[14] P. Cai, S. Wang, Y. Sun, and M. Liu, "Probabilistic end-to-end vehicle navigation in complex dynamic environments with multimodal sensor fusion," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4218–4224, Jul. 2020.

[15] F. Codevilla, M. Miiller, A. Ló pez, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–9.

[16] 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. Patt. Recognit.*, 2017, pp. 2174–2182.

[17] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," in *Proc. Mach. Learn. Res.*, vol. 87, Oct. 29–31, 2018, pp. 1–15.

[2]https://sites.google.com/view/vtgnet/

[18] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9329–9338.

[19] M. Bergqvist and O. Rödholm, "Deep path planning using images and object data," Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2018.

[20] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 772–788.

[21] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Proc. Robot.: Sci. Syst.*, FreiburgimBreisgau, Germany, Jun. 2019, doi: 10.15607/RSS.2019.XV.031.

[22] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Proc. Adv. Neu. Inf. Process. Syst.*, 1989, pp. 305–313.

[23] T. Wu, A. Luo, R. Huang, H. Cheng, and Y. Zhao, "End-to-end driving model for steering control of autonomous vehicles with future spatiotemporal features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 950–955.

[24] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, "High-speed autonomous drifting with deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1247–1254, Apr. 2020.

[25] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 435–453.

[26] S. Yang, W. Wang, C. Liu, and W. Deng, "Scene understanding in deep learning-based end-to-end controllers for autonomous vehicles," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 49, no. 1, pp. 53–63, Jan. 2018.

[27] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[28] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 793–803.

[29] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" *Adv. Neu. Inf. Process. Syst.*, pp. 5574–5584, 2017.

[30] L. Tai, P. Yun, Y. Chen, C. Liu, H. Ye, and M. Liu, "Visual-based autonomous driving deployment from a stochastic and uncertainty-aware perspective," *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 2622–2628, 2019.

[31] T. Fan, P. Long, W. Liu, J. Pan, R. Yang, and D. Manocha, "Learning resilient behaviors for navigation under uncertainty environments," 2019. [Online]. Available: https://arxiv.org/pdf/1910.09998.pdf

[32] M. Green, "How long does it take to stop?" Methodological analysis of driver perception-brake times," *Transp. Hum. Factors*, vol. 2, no. 3, pp. 195–216, 2000.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2018, pp. 4510–4520.

[34] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, "Intentionnet: Integrating planning and deep learning for goal-directed autonomous navigation," in *Proc. 1st Ann. Conf. Robot Learn.*, Nov. 13–15, 2017, vol. 78, pp. 185–194.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, Conference Track Proceedings, May 7–9, 2015.

[36] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 261–268.

[37] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

Yuxiang Sun received the bachelor's degree from the Hefei University of Technology, Hefei, China, in 2009, the master's degree from the University of Science and Technology of China, Hefei, China, in 2012, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2017. He is currently a Research Associate with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. His current research interests include autonomous driving, deep learning, robotics and autonomous systems, semantic scene understanding, etc. He was the recipient of the Best Paper in Robotics Award at IEEE-ROBIO 2019, and the Best Student Paper Finalist Award at IEEE-ROBIO 2015.

Hengli Wang received the bachelor's degree in mechatronics engineering from Zhejiang University, Hangzhou, China, in 2018. He is currently working toward the Ph.D. degree with the Robotics Institute, Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong. His research interests include computer vision, robot navigation, and deep learning.

Ming Liu (Senior Member, IEEE) received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005, and the Ph.D. degree from the Department of Mechanical and Process Engineering, ETH Zurich, Zurich, Switzerland, in 2013, supervised by Prof. Roland Siegwart. During his master's study with Tongji University, he stayed one year with the Erlangen-Nurnberg University and Fraunhofer Institute IISB, Erlangen, Germany, as a Master Visiting Scholar. He is currently with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, as an Associate Professor. His research interests include dynamic environment modeling, deep-learning for robotics, 3-D mapping, machine learning, and visual control. He was the recipient of second place in the European Micro Aerial Vehicle Competition (EMAV'09) and received two awards from International Aerial Robot Competition (IARC'14) as a Team Member, the Best RoboCup Paper Award for IROS 2013 (IEEE/RSJ International Conference on Intelligent Robots and Systems), and the Best Conference Paper Award for IEEE-CYBER 2015. He was the Best Student Paper Finalist for RCAR 2015 (IEEE International conference on Real-time Computing and Robotics), and the Best Student Paper Finalist for ROBIO 2015. He received the Best Student Paper Award for IEEE-ICAR 2017, the Best Paper in Automation Award for IEEE-ICIA 2017, the innovation contest Chunhui Cup Winning Award in both 2012 and 2013, and the Wu Weijun AI Award in 2016. He was the Program Chair of IEEERCAR 2016 and the Program Chair of International Robotics Conference in Foshan 2017. He was the Conference Chair of ICVS 2017. He was the awardee of IEEE IROS Toshio Fukuda Young Professional Award 2018.

Peide Cai received the bachelor's degree in automation from the School of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2018. He is currently working toward the Ph.D. degree with the Robotics Institute, Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong. His research interests include robot navigation, machine learning and sensor fusion.