See the Future: A Semantic Segmentation Network Predicting Ego-Vehicle Trajectory With a Single Monocular Camera

Yuxiang Sun, *Member, IEEE*, Weixun Zuo, and Ming Liu, *Senior Member, IEEE*

Abstract—**Ego-vehicle trajectory prediction is important for autonomous vehicles to detect collisions and accordingly avoid accidents. Recent approaches employ prior-known or on-line acquired road topology or geometries as motion constraints for their predictive models. However, the prior-known information (e.g., pre-built maps) might become unreliable due to, for example, temporal changes caused by road constructions. Whereas on-line perception may require high-cost sensors, such as large filed-of-view laser scanners, to get an overview structure of the local environment, making the prediction difficult to afford, especially for driving assistance systems. So in this letter, we provide a solution without using road topology or geometries for ego-vehicle trajectory prediction. We formulate this problem as a two-class semantic segmentation problem and develop a novel sequence-based deep neural network to predict the trajectory. The only sensor we need during runtime is a single front-view monocular camera. The inputs to our network are several consecutive images, and the output is the predicted trajectory mask that can be directly overlaid on the current front-view image. We create our datasets with different prediction horizons from KITTI. The experimental results confirm the effectiveness of our approach and the superiority over the baselines.**

Index Terms—**Trajectory prediction, ego-vehicle, semantic segmentation, ADAS, autonomous vehicles.**

I. Introduction

RECENT decades have witnessed the prosperity of Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. Ego-vehicle trajectory prediction is an important task for them. It estimates the vehicle positions several seconds into the future (i.e., the so-called prediction horizon). In the context of ADAS, the predicted trajectory could be integrated with obstacle detection algorithms to forecast and alert potential collisions. In the context of autonomous vehicles, ego-vehicle trajectory prediction and obstacle detection could be further integrated with autonomous control mechanisms to realize active collision avoidance.

Assuming that human drivers obey traffic rules and follow the road flawlessly [1], many recent approaches for ego-vehicle trajectory prediction employ prior-known or on-line acquired road geometries or topology as motion constraints for their predictive models [2]–[6]. However, this assumption might not always be satisfied in real traffic scenarios due to the uncertain intentions of human drivers. Moreover, in urban environments prior-known road information could become unreliable. For instance, temporal changes of road geometries caused by road constructions may be not updated timely in pre-built maps. Even though this issue could be addressed, the accuracy of the widely used GPS positioning systems are prone to be degraded in urban environments [7], making vehicles difficult to locate themselves in the maps. Some methods resort to on-line environmental perception, such as building local maps with Simultaneous Localization and Mapping (SLAM) techniques. However, on-line mapping may require deploying high-cost sensors, such as large field-of-view laser scanners, on vehicles to get a global overview for the structure of the local environment [6], making the system difficult to afford, especially for ADAS. In addition, the performance of on-line mapping might suffer from dynamic objects, such as moving vehicles or pedestrians in traffic environments. Robustly building maps in dynamic environments is still an open problem [8]–[10]. Low-cost visual sensors, such as monocular cameras, have been successfully used for SLAM tasks, but visual SLAM might be less suitable for this application due to not only the dynamic-environment issue but also the limited filed-of-view.

Different from the previous work, we provide a solution for ego-vehicle trajectory prediction without using road topology or geometries. Moreover, unlike most of approaches that represent trajectories on bird-view maps, we overlay trajectories on 2-D front-view images as shown in Fig. 1. As we can see, the trajectory in this work is not represented as lines in 2-D or 3-D maps. It is represented as car-width pixel-wise trajectory labels projected from the 3-D space to the 2-D image plane [11]. With such data representation, human-drivers could see the future trajectory intuitively on a Head Up Display (HUD). More importantly, this kind of data representation allows us to formulate the prediction problem as an end-to-end two-class semantic segmentation problem (i.e., the trajectory class and background class), so that we can take as input only the raw images streamed by a single front-view camera and output the predicted trajectory masks. In this letter, we develop a novel sequence-based deep neural network to achieve this goal.

Manuscript received August 14, 2019; accepted January 23, 2020. Date of publication February 20, 2020; date of current version March 4, 2020. This letter was recommended for publication by Associate Editor I. Manchester and Editor J. Roberts upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China Project No. U1713211, in part by the Guangdong-Hong Kong Cooperation Innovation Platform Project No. 2018B050502009, and in part by the Research Grant Council of Hong Kong Government Project No. 11210017 and 21202816. *(Corresponding author: Ming Liu.)*

The authors are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: eeyxsun@ust.hk, sun.yuxiang@outlook.com; wzuo@connect.ust.hk; eelium@ust.hk).

Digital Object Identifier 10.1109/LRA.2020.2975414

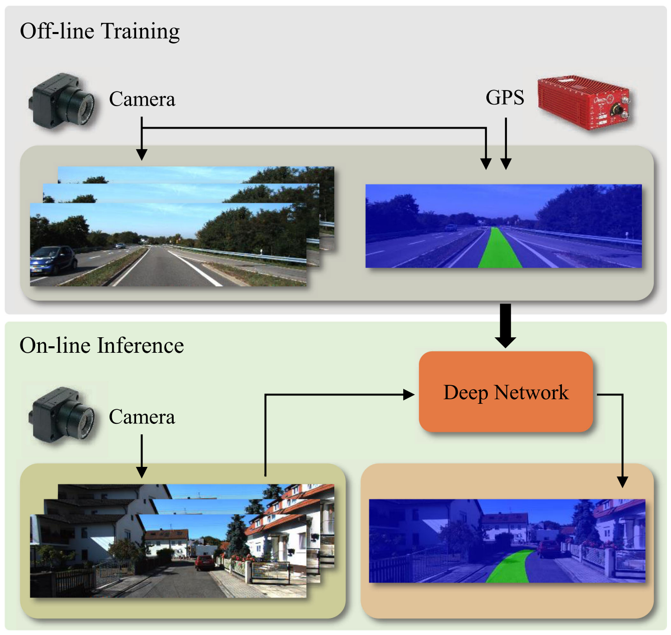

Fig. 1. The overall pipeline of our approach. At the off-line training stage, we need a monocular camera and a GPS positioning system. At the on-line inference stage, we only need a monocular camera. We formulate the trajectory prediction problem as a two-class semantic segmentation problem. The two classes are coloured as blue (background) and green (trajectory), respectively. The predicted trajectory could be displayed on a head-up display, allowing drivers to see the future straightforwardly. The shown prediction horizon in this figure is 3 s. The figure is best viewed in color.

The main idea of designing our network is that we believe the future trajectory could be inferred from the current and past visual information. So we take as input several consecutive front-view images, and supervisely train the network with the current ground-truth trajectory mask. The experimental results confirm the effectiveness of our idea. We summarize the main contributions of this letter as follows:

1) We develop a novel end-to-end sequence-based deep semantic segmentation network that predicts ego-vehicle trajectories with a single monocular camera.
2) We create datasets with different prediction horizons from the KITTI dataset [12]. The evaluation results confirm the effectiveness of our approach.
3) We build baselines based on convolutional LSTM [13] and Deeplab v3+ [14]. The comparative results demonstrate the superiority of our network.

II. Related Work

A. Trajectory Prediction

As aforementioned, many methods on ego-vehicle trajectory prediction use road information for their predictive models. Their mainstream predictive models are built on vehicle physical states (e.g., position, velocity, yaw rate, acceleration) or human-diver maneuvers (e.g., brake, speed up, turn left or right). Acquiring these information usually needs vehicle dynamic sensors (e.g., wheel speed sensor, steering angle sensor, brake pressure sensor, yaw rate and acceleration sensors) or exteroceptive sensors (e.g., Lidar, RTK GPS). Moreover, maintaining synchronization between different sensors needs extra devices and efforts. Our method that relies only a monocular camera is a low-cost solution, and allows the prediction system to be deployed easily.

Houenou *et al.* [2] built local parabolic models for road center lines by visual detections of road markings. A maneuver recognition module was designed for long-term trajectory prediction by combining the road information and the vehicle physical states. Kim *et al.* [3] measured vehicle physical states and road geometries from a commercial RTK GNSS/INS system and a commercial Mobileye visual system. They developed two Kalman filters to recursively estimate the physical states and road geometries. Note that vision-based road geometry estimation suffers from the occlusions caused by other traffic participants. Guo *et al.* [4] predicted the longitudinal trajectory in the Frenet frame. They classified vehicle states into non-maneuvering and maneuvering. For the non-maneuvering sate, the constant acceleration model was used with the physical states to predict trajectories. For the maneuvering state, the quintic polynomial model was employed with the context information to predict trajectories. Raipuria *et al.* [5] modelled vehicle motion in a function of road structures. They used 6 laser scanners to get the relative vehicle position, velocity and heading angles. The relative positions obtained from the laser scanners are transformed to prior-known maps via GPS. The road geometries can be inferred from the coefficients of the qubic functions.

There are also works that predict trajectories of other traffic participants on a moving vehicle. One of the seminal works was proposed by Geiger *et al.* [15]. They employed various visual cues to infer 3-D layout of traffic scenes as well as vehicle locations and orientations. Recently, Kim *et al.* [16] proposed to use the Recurrent Neural Network (RNN) to predict the future positions, which were represented probabilistically on local grid maps.

B. Semantic Segmentation

Badrinarayanan *et al.* [17] introduced the Encoder-Decoder architecture that is widely used nowadays. Chen *et al.* [14] proposed Deeplab v3+ by taking their previous Deeplab v3 as the encoder. Specifically, they incorporated the dilated convolutions into a feature extractor backbone (e.g., ResNet) and a spatial pyramid pooling module [18]. The encoder outputs a low-level feature map and a high-level feature map, which are concatenated and up-sampled in a decoder. Luc *et al.* [19] proposed a batch-based method and an autoregressive method to predict more than one video frames and their semantic labels. The former one predicts all future frames at once, while the latter makes predictions sequentially. Instead of first estimating future frames then semantic labels like [19], Chiu *et al.* [20] observed that future frames are not necessary for future semantic reasoning. So they proposed an end-to-end network that takes as input several preceding frames and directly outputs the semantic labels of the next frame.

C. Trajectory Prediction via Semantic Segmentation

Baumann *et al.* [6] used a Lidar to build local grid maps of the surrounding static environment, which were then converted to bird-view gray-scale images. The past path estimated from Lidar odometry were plotted on binary bird-view images. They concatenated the two types of images and fed it into a semantic segmentation network (e.g., SegNet [17]). The network was trained with future trajectory images. The work [6] and ours mainly differ in three-fold. Firstly, path in [6] is represented in bird-view grid maps, while our method projects car-width future trajectory on the current front-view image. Secondly, [6] requires road topology including lanes as input, which are encoded by local grid maps built with a Lidar, while our method does not require such information. We only need a low-cost camera during runtime. Lastly, [6] is not end-to-end. Lidar scans are firstly employed to compute local grid maps and path (odometry), which serve as the input to the model. The predicted path is also not directly obtained from the output. The model firstly outputs a prediction map. Then a mechanism was developed to extract the predicted path from the prediction map. While our method is end-to-end, it directly takes as input raw front-view camera images and outputs the predicted trajectory overlaid on the current image.

Barnes *et al.* [21] and Zhou *et al.* [22] proposed drivable path segmentation methods with front-view images. They assumed that human-driven paths are drivable and derived the car-width ego-vehicle path using the odometry from stereo vision or IMU/wheel encoder. The future ego-vehicle path was projected on the current front-view image and refined with obstacle detection, which was used as the ground-truth image. They trained the off-the-shelf single image-based semantic segmentation networks, such as SegNet [17] and ENet [23], given the current single front-view image as input. We believe that the work of [21] and [22] can be also seen as ego-vehicle trajectory prediction, because they both used the future trajectory as the ground truth. The major difference between ours and [21] or [22] would be that our method is sequence-based, while they are single image-based. To the best of our knowledge, ours is the first sequence-based semantic segmentation work for ego-vehicle trajectory prediction.

III. The Proposed Approach

A. The Approach Overview

Fig. 1 shows the two stages of our overall pipeline: the off-line training stage and the on-line inference stage. During the training stage, we employ camera and GPS to generate training images. Specifically, we use the odometry information from a GPS positioning system to plot future car-width trajectory on the current image, which serves as the ground truth for training. Although we use GPS, odometry computed with other sensor data can be replaced here [24]. During the on-line inference stage, we only need a single camera to stream the front-view images. The input to our network is a set of three consecutive images captured at time $t-2$, $t-1$ and t. The output is the predicted trajectory mask that can be directly overlaid on the current image captured at time t.

B. Generating Ground-Truth Car-Width Trajectory Images

This work uses the KITTI dataset [12], because the positions of the Wheel-Ground Contact (WGC) points in the vehicle coordinate are publicly given. We define the trajectory as the filled area bounded by the time-evolving WGC points of the two front wheels. We project the future trajectory on the current front-view image as the ground truth.

Let g_0 denote the GPS coordinate frame at the start; g_t and c_t denote the GPS and camera frames at current time t, respectively; g_{t+k} denote the future GPS frame at time $t+k$; p_{t+k} denote the 3-D WGC points at time $t+k$. The task is to project the 3-D point p_{t+k} to 2-D pixel coordinate I in camera frame c_t, which can be calculated by:

$$^{c_t}I_{p_{t+k}} = \underbrace{K \times {}^{c_t}{}_{g_t}T}_{\text{Calibration}} \times \underbrace{({}^{g_0}{}_{g_t}T)^{-1} \times {}^{g_0}{}_{g_{t+k}}T}_{\text{GPS}} \times \underbrace{{}^{g_{t+k}}p_{t+k}}_{\text{Calibration}} \quad (1)$$

where K, ${}^{c_t}{}_{g_t}T$ and ${}^{g_{t+k}}p_{t+k}$ respectively represent the camera intrinsic matrix, the transformation from the GPS frame to the camera frame, the coordinate of the WGC point with respect to the GPS frame. They can be considered as constant variables in each day of the KITTI dataset. We get them from the KITTI calibration files. ${}^{g_0}{}_{g_t}T$ and ${}^{g_0}{}_{g_{t+k}}T$ respectively represent the transformations from the current and future GPS frames to the start point. They are read from the GPS odometry. The projected pixels are formed as convex polygons and then filled as the car-width trajectory. The parameter k in (1) is measured in milliseconds. We adjust different k to generate different datasets with different prediction horizons.

C. The Proposed End-to-End Sequence-Based Network

Fig. 2 displays the overall architecture of our proposed end-to-end prediction network. We need three consecutive front-view images (i.e., $t-2$, $t-1$ and t) for each inference.

Firstly, two consecutive images are taken as input for an optical flow network to extract motion features of the ego-vehicle. We use FlowNet 2.0 [25] in this work. However, any network with the ability to generate optical flow can be embedded here. The motion features are fused with the current image from time t (taking $t-1$ and t as an example) by tensor concatenation.

Secondly, the fused feature map is pixel-wisely converted to embedding vectors through a backbone feature extraction module and an embedding head. In this work, we borrow the Atrous CNN and Spatial Pyramid Pooling (ASPP) from Deeplab v3+ [14] to serve as the backbone feature extraction module. The extracted base features from the backbone feature extraction module are sent to the embedding head, which consists of a single convolution layer. The structure of the embedding head is shown in the right part of Fig. 2.

Thirdly, we fuse the embedding vectors from $t-2$ and $t-1$ (named as the previous embedding vectors) with the embedding vectors from $t-1$ and t (named as the current embedding vectors) by element-wise addition. Then, the fused embedding vectors are sent to a matching module to compute matching scores, which measure the similarity between the previous and current embedding vectors. The structure of the matching module is shown in the right part of Fig. 2. It consists of a convolutional layer and a Sigmoid layer. The output of the matching module is a single-channel feature map (with the same resolution of the base feature map), in which each element ranges from 0 to 1.

Lastly, the base feature map is weighted by the matching scores through element-wise multiplication. The weighted base features are sent to a convolutional block (named as Interpretation block) to interpret the visual trajectory. The configurations of the convolutional layers are similar to those of Deeplab v3+ decoder. Then, the visual trajectory feature map is up-sampled through bilinear interpolation to the original resolution. A soft-max layer is added as the last layer for training.

Note that all the modules in our network, including the backbone feature extraction module, the embedding head, the matching module and the interpretation block, share their individual weights at different times. In addition, we infer the trajectory at

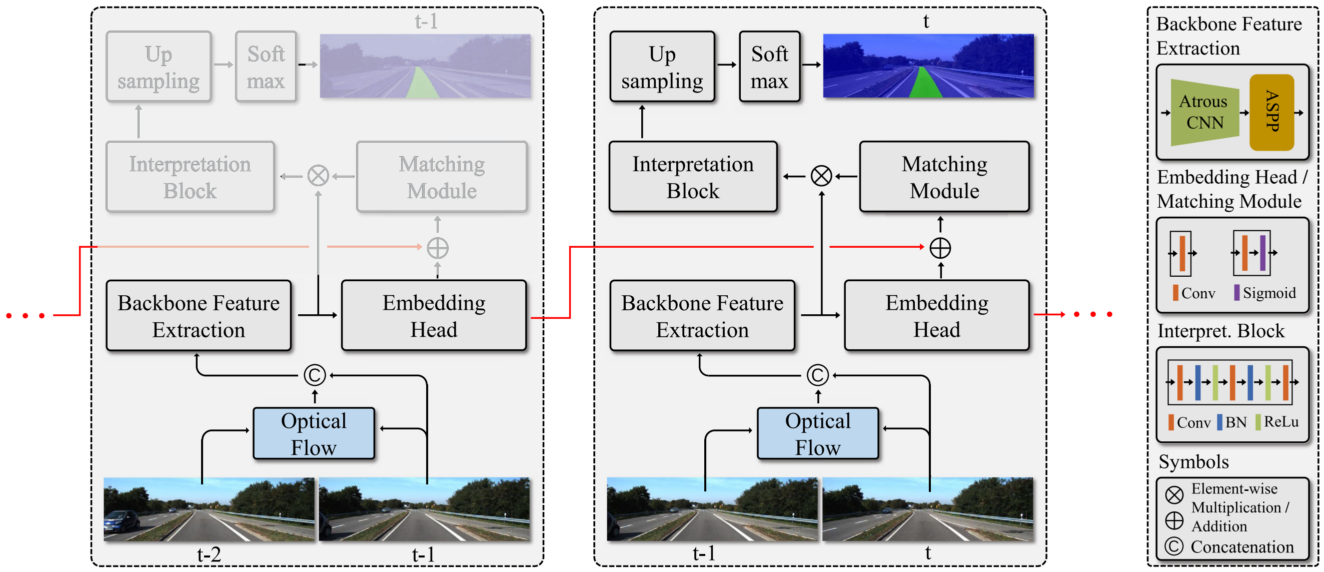

Fig. 2. The structure of our proposed end-to-end sequence-based deep network for trajectory prediction. The detailed structures of each module are displayed in the right part of the figure. Our network takes as input three consecutive front-view images from $t-2$ to t, and outputs the predicted trajectory mask that can be overlaid on the current image at time t. We make the matching module, the interpretation block, the up-sampling layer and the softmax layer in the left box opaque to indicate that the trajectory is only predicted at time t. The figure is best viewed in color.

time t, so the predicted trajectory mask is overlaid on the current front-view image at time t.

IV. The Experimental Results and Discussions

A. *The Generated Dataset*

We use the *City*, *Road* and *Residential* sequences (recorded with the camera No. 2) from the KITTI raw dataset to create our dataset. The *Person* and *Campus* sequences are not used because they are not real driving scenarios or the length of the sequences are too short. We require that the length of sequences can at least support 5 s prediction horizon. So some short sequences in the *City*, *Road* and *Residential* categories are also not used. We select sequences with relative good GPS signals from the three categories, because we need good GPS signals to obtain reliable odometry to draw ground-truth trajectory images. Our dataset is split into *train* (15578 frames), *validation* (3980 frames) and *test* (3922 frames), taking the ratios around 66%, 17% and 17%, respectively. In the *train* set, the numbers of the selected training images from the three categories (*City*, *Road* and *Residential*) are 1632, 1897, 12049, respectively. Note that we use entire sequences for training, validation and testing. In other words, all the images in a sequence are used either for training or validation or testing. So the validation and testing images are ensured not from the sequences that are used for training. We also checked that the driving routes, driving behaviours and the visual appearances along driving are not the same between training and testing, which could avoid over-fitting to a particular driving scenario. To evaluate the prediction performance for different horizons, we made 5 datasets with different prediction horizons ranging from 1 s to 5 s. The number of images in the 5 datasets varies a little bit, because the images in the last prediction horizon in each sequence are abandoned due to the unavailability of the future GPS data.

B. *Training Details*

We implement our network using PyTorch 1.1 with CUDA 10.1 and cuDNN 7.0 libraries. Our network is trained on a PC with an Intel i7 CPU and an NVIDIA 1080 Ti graphics card. We resize all the images from the original resolutions (i.e., 375×1242, 370×1226 and 376×1241) to 193×640 for efficiency and accordingly adjust the batch size to fit for the 11GB graphics memories. For the Atrous CNN, we use the ResNet-101 as the backbone. We train our network with the pre-trained weight of ResNet-101 provided by PyTorch. Other layers in our network are initialized with the Kaiming normal scheme [26]. For the FlowNet 2.0, we only use the forward inference with the given pre-trained weight.

We employ the Stochastic Gradient Descent (SGD) [27] as our optimization solver, in which the initial learning rate, momentum and weight decay are set to 0.01, 0.9 and 0.0005, respectively. The input training images are randomly shuffled in each sequence before training. Note that the shuffle does not influence the order of the 3 consecutive images in each input image set. It shuffles between image sets. The learning rate is decayed exponentially at each epoch. We train the network until the validation loss converges. As the pixels of the two classes (i.e., background, predicted trajectory) are out of balance in our dataset, we use the focal loss [28] to down-weight the contributions from the easy class and focus the model on the hard one.

C. *Ablation Study*

1) Ablation for Optical Flow: We firstly check whether the optical flow module benefits our trajectory prediction. To this end, we make a variant of our network without optical flow. The input to this variant is the concatenation of two consecutive images. We name this variant as NOF (NO optical Flow). To observe the impacts caused by different optical flow networks,

TABLE I
THE ABLATION STUDY RESULTS FOR DIFFERENT OPTICAL FLOW ALGORITHMS. WE COULD FIND THAT USING FLOWNET 2.0 OUTPERFORMS THE OTHERS. MOREOVER, THE RESULTS SHOW THAT USING OPTICAL FLOW WOULD BE A BENEFIT HERE

	Ours	TVF	HD3F	NOF
Acc	91.64%	91.05%	90.81%	88.06%
IoU	74.01%	71.50%	69.82%	66.24%

TABLE II
THE ABLATION STUDY RESULTS FOR DIFFERENT FUSION STRATEGIES. WE COULD FIND THAT OUR FUSION STRATEGY GIVES THE BEST PERFORMANCE IN GENERAL

	Ours	FASC	BC	BA
Acc	91.64%	91.10%	91.39%	91.72%
IoU	74.01%	70.96%	70.56%	64.37%

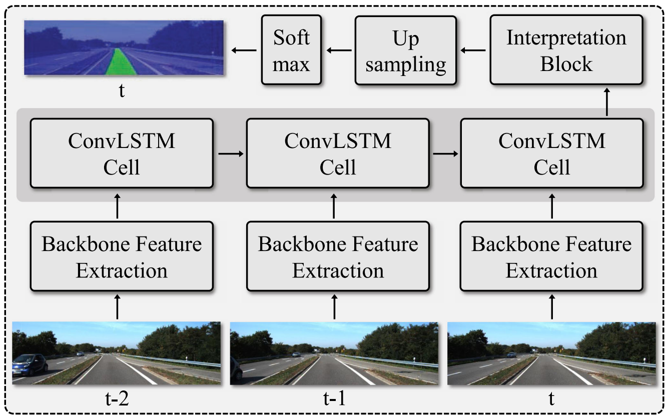

Fig. 3. The network structure of the convolutional LSTM baseline. *Conv* stands for *Convolutional*. The LSTM network consists of three convolutional LSTM cells, which take as inputs the images at $t-2$, $t-1$ and t, respectively. The structures of the *Backbone Feature Extraction* module and *Interpretation Block* are shown in Fig. 2.

we compare two variants of our network using TVNet [29] and HD3Net [30], respectively. They are named as TVF (TVNet optical Flow) and HD3F (HD3Net optical Flow). We think that the 3 s prediction horizon could be representative because of the average 1.5 s for humans to be able to react and avoid an accident [31]. The 3 s prediction horizon is chosen to double the reaction time [32] considering the vehicle speed. So we train and test all the variants on the dataset with the 3 s prediction horizon in this section.

In this work, we use the Acc and IoU metrics that are defined in [33] to measure the segmentation accuracy of the predicted trajectory. Table I displays the comparative results. As we can see, the NOF variant presents the worst performance, indicating that using optical flow would be able to boost the performance for our network. The reason might be that optical flow provides well described ego-motion features, which helps to detect the movement tendency of the ego-vehicle. From the comparison, we also find that our network using FlowNet 2.0 presents the best performance.

2) Ablation for Fusion Strategies: There are two fusion operations in our network. The first is the fusion of image and optical flow data. The second is the fusion of the previous and current embedding vectors. In our network, we use concatenation and element-wise addition for the two fusion operations, respectively. To validate the effectiveness of this fusion strategy, we create three variants: 1) First Addition and Second Concatenation (FASC); 2) Both Concatenation (BC); 3) Both Addition (BA). Note that the number of channels for the optical flow and image data are 2 and 3, respectively. To allow the element-wise addition for the optical flow and image data in the BA variant, we apply a convolutional layer on the image data to reduce the number of channels to 2. The comparative results are shown in Table II. We find that the Acc values are not sensitive to different fusion strategies, but IoU exhibits notable degradations for other variants. This indicates that the false positives increase significantly. Many pixels from the background are incorrectly classified as the predicted trajectory. The reason might be that the concatenation operation is more suitable for heterogeneous data (e.g., flow and image), while the addition is more suitable for the homogeneous data (e.g., previous and current embedding vectors). The comparative results demonstrate the superiority and confirm the effectiveness of our fusion strategy.

D. Baseline Methods

1) The Baseline Based on Convolutional LSTM (ConvLSTM Baseline): Long Short Term Memory (LSTM) networks [34] are designed to process sequential information, and has been successfully used for trajectory prediction. We build a baseline based on LSTM. The network structure is shown in Fig. 3. We firstly extract the backbone features from the input. Then, the extracted feature maps are sent to a LSTM network that consists of three convolutional LSTM cells [13]. Thirdly, the hidden states of the last LSTM cell are sent to the Interpretation Block and Up-sampling module. We finally apply a softmax layer on the feature maps to get the output. To ensure fair comparison, we use the same *Backbone Feature Extraction* module, *Interpretation Block* and *Up-sampling* module that are used in our network, except modifications for the channel numbers. Note that the reason why we use convolutional LSTM is that convolutional LSTM replaces the fully connected layers in the vanilla LSTM with convolutional layers, so the input to LSTM accepts feature maps, allowing the baseline to avoid extensive loss of spatial information.

2) The Baseline Without Sequential Information (No-Seq Baseline): To check whether the sequential information (i.e., the consecutive images) benefits the prediction, we build a baseline without using sequential information. We train a semantic segmentation network given merely the current image from time t (not the three consecutive images). Here we use Deeplab V3+ [14]. This baseline can be seen as an enhanced version of [21] and [22], because Deeplab V3+ is in general more powerful than SegNet [17] and ENet [23].

E. Comparative Results

1) Quantitative Results: We compare our network with the baselines on the datasets with different prediction horizons (i.e., 1 s–5 s). Table III displays the quantitative comparative results. In general, we can see that our network achieves the best performance across all the prediction horizons, which demonstrates the superiority of our network. Especially, we find that our network significantly outperforms the baselines on the IoU metric. Comparing ours with the ConvLSTM baseline, there

TABLE III
THE COMPARATIVE RESULTS ON THE DATASETS WITH DIFFERENT PREDICTION HORIZONS. BEST RESULTS ARE HIGHLIGHTED IN BOLD FONT. OUR NETWORK ACHIEVES THE BEST PERFORMANCE OVER ALL THE PREDICTION HORIZONS

Methods	1s		2s		3s		4s		5s	
	Acc	IoU	Acc	IoU	Acc	IoU	Acc	IoU	Acc	IoU
Ours	**97.09%**	**72.85%**	**94.31%**	**73.05%**	**91.64%**	**74.01%**	**90.71%**	**72.12%**	**89.00%**	**72.25%**
ConvLSTM Baseline	93.29%	64.14%	84.53%	67.21%	79.81%	65.80%	76.19%	56.47%	73.18%	56.00%
No-Seq Baseline	92.59%	61.04%	92.53%	60.06%	90.74%	59.07%	86.30%	57.87%	85.36%	62.48%

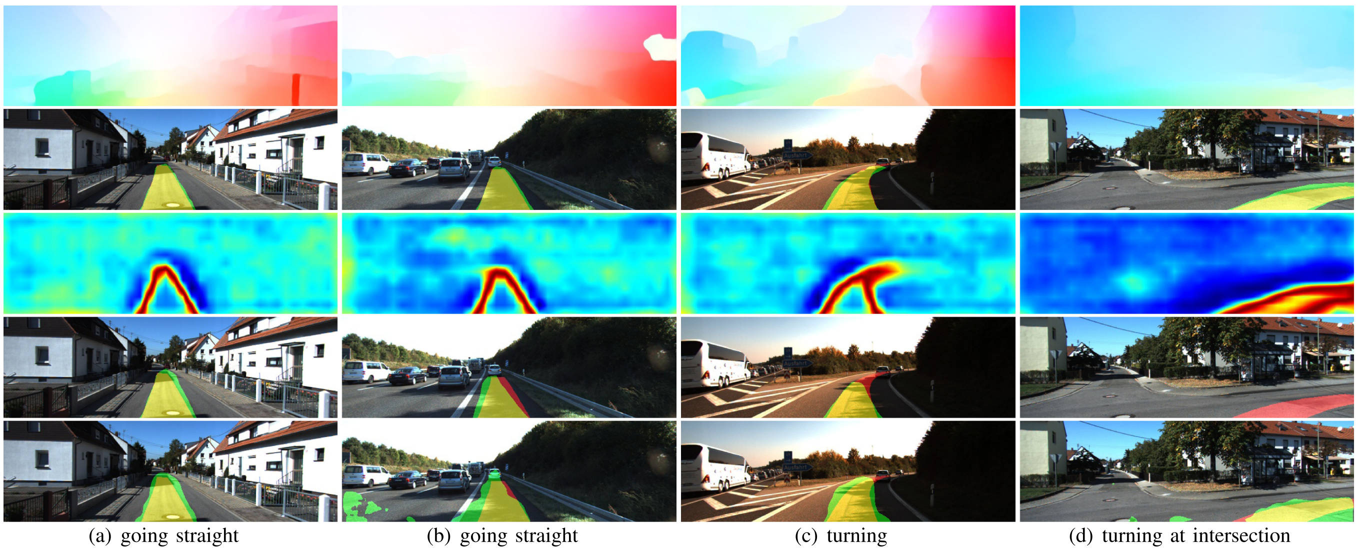

Fig. 4. Some acceptable prediction examples. The top to bottom rows respectively show the optical flow computed from the last and the current images, our results, the uncertainty estimation of our results, the ConvLSTM baseline results and the No-Seq baseline results. The red, green and yellow colors in the trajectory images respectively denote the ground truth, the prediction results and the intersection between them. The uncertainties increase from blue to red. The figure is best viewed in color.

could be less true positives or more false negatives leading to the lower IoU for the ConvLSTM baseline, because our network also surpasses it significantly on the Acc metric. For the No-Seq baseline, as it achieves comparable Acc values with ours, we think that there could be more false positives leading to the lower IoU. We conjecture that the absence of sequential information in the No-Seq baseline makes the network feel like doing a road segmentation task, hence causing many false positives.

The comparative results show that our paradigm for sequential information processing would be more suitable than LSTM in this task. In addition, the sequential information (i.e., the consecutive images) is necessary for prediction, otherwise the network could treat the prediction task as a normal semantic segmentation task and hence generate wrong predictions. We observe from Table III that the prediction capabilities of all the networks decrease with the increased prediction horizon. This is expected because it would not make sense to predict very long horizons without given other cues or prior-known information. Nevertheless, we find that even in the worst case (i.e., 5s) our network still gives around 90% Acc and 70% IoU, indicating that our network would have the capability to predict relative long horizons. For the ConvLSTM baseline, it works for shorter horizons, but the performance drops notably when increasing the horizon, making it not suitable for predicting long horizons.

2) Qualitative Demonstrations: The qualitative experiments are performed on the dataset with 3 s prediction horizon. Fig. 4 displays some acceptable results of our network and the comparative results from the baselines. As we can see, our network predicts the trajectory with the minimum false positives. From the sub-figures (c) and (d), we find that the ConvLSTM baseline may have weak capability for predicting turning cases. Especially from (d), the ConvLSTM baseline totally fails to predict the trajectory when turning at an intersection. From the sub-figures (b) and (d), we find that there are many background pixels wrongly classified as the trajectory (false positives) in the results of the No-Seq baseline. Particularly in (b), even the pixels from other lanes are labelled as the trajectory, indicating that without using the sequential information the No-Seq baseline might tend to treat the task as a normal road segmentation task.

Fig. 5 demonstrates some unacceptable results of our network and the comparative results from the baselines. The sub-figure (d) shows an intersection-turning case. Comparing Figs. 4(d) and 5(d), we think the reason for our unacceptable prediction is that the lateral movement of the vehicle at the moment has not reached the extent sufficient for the prediction. This could be observed from the optical flow results. In Fig. 4(d), the horizontal components with almost one direction (coloured as blue) dominate the optical flow map, indicating that the vehicle is behaving large lateral movement at that moment. While the optical flow map of Fig. 5(d) resembles the one that is going straight (e.g., left blue and right red), there is little lateral movement so the network fails to predict. Fig. 5(b) and (c) show the cases that the

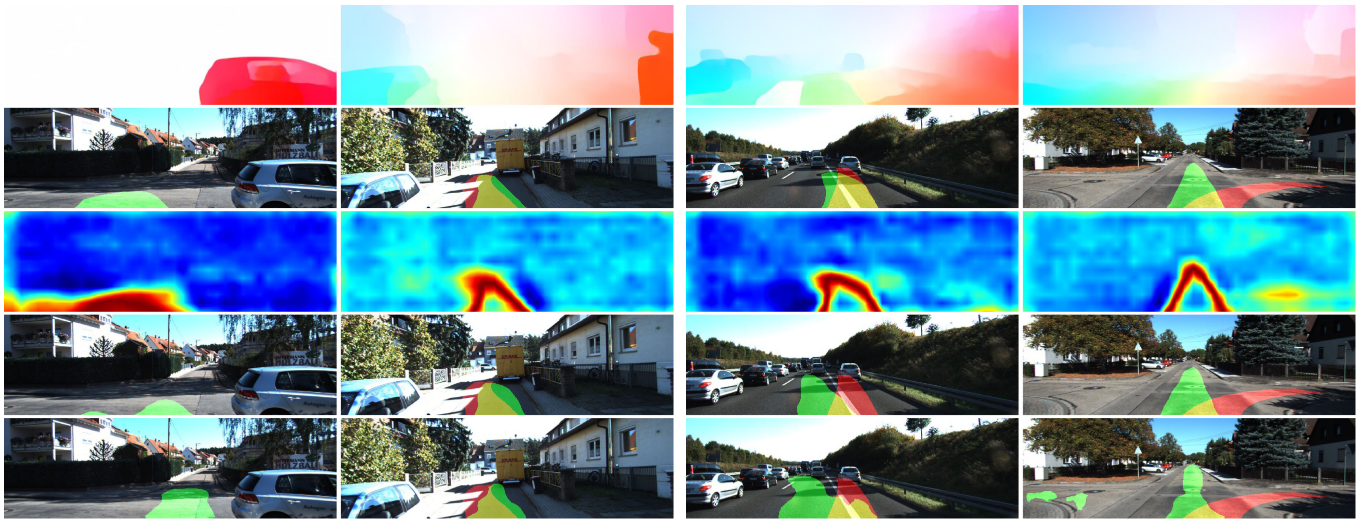

Fig. 5. Some unacceptable prediction examples. The descriptions for the top to bottom rows, as well as the colors in the trajectory images and uncertainly maps can be found in the caption of Fig. 4. The figure is best viewed in color.

vehicle is changing lane, we can see that our network is only able to predict the majority of the trajectory, but our results are better than the baselines in terms of the false positives, especially in Fig. 5(c) the baselines seem to fail the prediction. Fig. 5(a) shows a case that the vehicle is stopping at an intersection. In such a case, the correct prediction is no output since the vehicle will not move in the future. The optical flow values of the background approach zero (coloured as white). We think that the reason for our false output comes from the disturbances of the optical flow of the moving object (i.e., the front moving vehicle), because we observe that our network does not give output when the moving vehicle is not in the field-of-view. However, the baselines always output false positives no matter whether there exists the moving vehicle. The supplementary video for the qualitative demonstrations is shown here.[1]

As the semantic segmentation gives deterministic inference, we use the Monte Carlo (MC) dropout technique [35] to estimate the uncertainties of the segmentation results, which could be seen as a probabilistic representation for the trajectory prediction. Specifically, we insert several dropout layers in our network. During testing, we run interference 50 times for each prediction. The uncertainty is calculated using the entropy function [36] with the averaged softmax outputs over the 50 inferences. The uncertainty maps are shown in Figs. 4 and 5. In Fig. 5(a), although our network gives the incorrect prediction, we can see that the model is actually really not sure about its prediction. So the uncertainty map can be used as a probabilistic indicator to compensate the negative impacts caused by potential wrong predictions.

3) Inference Time: We evaluate the inference time on an NVIDIA 1080Ti with the input resolution of 193×640. Table IV displays the results. As our method consists of our proposed network and optical flow, the total time cost would be $21.56 + 2 \times 9.41 = 40.38$ ms, where 9.41ms is the time cost for one optical flow inference and we have two in our network. We can see that our method runs faster than the ConvLSTM Baseline, though they both take as input three images. The No-Seq baseline is much faster partially because it only processes one image during each inference.

[1] https://github.com/yuxiangsun/See-the-Future

TABLE IV
THE INFERENCE TIME FOR THE NETWORKS. THE TESTED HARDWARE IS AN NVIDIA 1080TI GRAPHICS CARD. THE UNIT IS MILLISECOND (MS)

	Ours	ConvLSTM Baseline	No-Seq Baseline
Time (ms)	21.56 (+9.41 × 2)	51.61	10.71

F. The Limitations

We consider the major limitation of our network as the degraded prediction performance when turning at intersections. As our network inferences solely based on the input visual images, it can only make decisions when it sees enough lateral movements of the vehicle. Otherwise, the network tend to treat the movement as going straight and predict the trajectory forward. In addition, the way the ground truth is determined does not allow the model to learn long-term prediction, such as one minute or more prediction horizon. This is because at such level of prediction horizon, the vehicle motion cannot be largely determined from current ego-motion status (i.e., the natural inertia is not dominant), and the ground truth trajectories cannot cover all possible driven routes.

V. CONCLUSION

We proposed here a novel semantic segmentation network to predict the future trajectory of the ego-vehicle. Our network is end-to-end. It takes as input several consecutive raw images from a single front-view monocular camera, and outputs the predicted trajectory mask that can be directly overlaid on the current image. We create our datasets with different prediction horizons from KITTI. The experimental results confirm the effectiveness of our network design and the superiority over the baselines. In the future, we would like to integrate human intentions to improve the prediction performance when turning at intersections. In addition, we would like to generate more training data for lane-changing cases using simulation software to improve the prediction performance when changing lanes.

REFERENCES

[1] M. Schreier, V. Willert, and J. Adamy, “An integrated approach to maneuver-based trajectory prediction and criticality assessment in arbitrary road environments,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2751–2766, Oct. 2016.

[2] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, “Vehicle trajectory prediction based on motion model and maneuver recognition,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 4363–4369.

[3] B. Kim and K. Yi, “Probabilistic and holistic prediction of vehicle states using sensor fusion for application to integrated vehicle safety systems,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2178–2190, Oct. 2014.

[4] C. Guo, C. Sentouh, B. Soualmi, J.-B. Haué, and J.-C. Popieul, “Adaptive vehicle longitudinal trajectory prediction for automated highway driving,” in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2016, pp. 1279–1284.

[5] G. Raipuria, F. Gaisser, and P. P. Jonker, “Road infrastructure indicators for trajectory prediction,” in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2018, pp. 537–543.

[6] U. Baumann, C. Guiser, M. Herman, and J. M. Zollner, “Predicting ego-vehicle paths from environmental observations with a deep neural network,” in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 1–9.

[7] H. Huang, Y. Sun, and M. Liu, “Reliable monocular ego-motion estimation system in rainy urban environments,” in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2019, pp. 1290–1297.

[8] Y. Sun, M. Liu, and M. Q.-H. Meng, “Improving RGB-D SLAM in dynamic environments: A motion removal approach,” *Robot. Auton. Syst.*, vol. 89, pp. 110–122, 2017.

[9] J. Cheng, Y. Sun, and M. Q.-H. Meng, “Improving monocular visual slam in dynamic environments: An optical-flow-based approach,” *Adv. Robot.*, vol. 33, no. 12, pp. 576–589, 2019.

[10] Y. Sun, M. Liu, and M. Q.-H. Meng, “Motion removal for reliable RGB-D SLAM in dynamic environments,” *Robot. Auton. Syst.*, vol. 108, pp. 115–128, 2018.

[11] 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.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[13] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015.

[14] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 801–818.

[15] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3D traffic scene understanding from movable platforms,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 1012–1025, May 2014.

[16] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, “Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network,” in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst.*, 2017, pp. 399–404.

[17] V. Badrinarayanan and A. Kendall and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[18] Y. Sun, L. Wang, Y. Chen, and M. Liu, “Accurate lane detection with atrous convolution and spatial pyramid pooling for autonomous driving,” in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2019, pp. 642–647.

[19] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun, “Predicting deeper into the future of semantic segmentation,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 648–657.

[20] H. Chiu, E. Adeli, and J. C. Niebles, “Segmenting the future,” 2019, *arXiv:abs/1904.10666*.

[21] D. Barnes, W. Maddern, and I. Posner, “Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 203–210.

[22] W. Zhou, S. Worrall, A. Zyner, and E. Nebot, “Automated process for incorporating drivable path into real-time semantic segmentation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–6.

[23] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, “ENet: A deep neural network architecture for real-time semantic segmentation,” 2016, *arXiv:abs/1606.02147*.

[24] H. Huang, Y. Sun, H. Ye, and M. Liu, “Metric monocular localization using signed distance fields,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1195–1201.

[25] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 2462–2470.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 1026–1034.

[27] G. Montavon, G. Orr, and K.-R. Mller, *Neural Networks: Tricks of the Trade*, 2nd ed. Berlin, Germany: Springer, 2012.

[28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 2980–2988.

[29] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang, “End-to-end learning of motion representation for video understanding,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 6016–6025.

[30] Z. Yin, T. Darrell, and F. Yu, “Hierarchical discrete distribution decomposition for match density estimation,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2019.

[31] M. Green, ““How long does it take to stop?” Methodological analysis of driver perception-brake times,” *Transpo. Human Factors*, vol. 2, no. 3, pp. 195–216, 2000.

[32] C. Barrios and Y. Motai, *Predicting Vehicle Trajectory*. Boca Raton, FL, USA: CRC Press, 2017.

[33] Y. Sun, W. Zuo, and M. Liu, “RTFNet: RGB-thermal fusion network for semantic segmentation of urban scenes,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2576–2583, Jul. 2019.

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

[35] A. Kendall, V. Badrinarayanan, and R. Cipolla, “Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding,” 2015, *arXiv:/1511.02680*.

[36] P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun, “Efficient uncertainty estimation for semantic segmentation in videos,” in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 520–535.