IET Intelligent Transport Systems

Research Article

Multi-task Deep Learning with Optical Flow Features for Self-Driving Cars

ISSN 1751-956X
doi: 0000000000
www.ietdl.org

Yuan Hu[1], *Hubert P. H. Shum*[2*], *Edmond S. L. Ho*[3]

[1] *Department of Mechanical and Electrical Engineering, Henan Agricultural University, No. 63 Agricultural avenue, Zhengzhou City, China*
[2] *Department of Computer Science, Durham University, Durham, DH1 3LE, United Kingdom*
[3] *Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, NE1 8ST, United Kingdom*
* *E-mail: hubert.shum@durham.ac.uk*

Abstract: The control of self-driving cars has received growing attention recently. While existing research shows promising results in vehicle control using video from a monocular dash camera, there has been very limited work on directly learning vehicle control from motion-based cues. Such cues are powerful features for visual representations, as they encode the per-pixel movement between two consecutive images, allowing a system to effectively map the features into the control signal. We propose a new framework that exploits the use of a motion-based feature known as *optical flow* extracted from the dash camera, and demonstrates that such a feature is effective in significantly improving the accuracy of the control signals. Our proposed framework involves two main components. The flow predictor, as a self-supervised deep network, models the underlying scene structure from consecutive frames and generates the optical flow. The controller, as a supervised multi-task deep network, predicts both steer angle and speed. We demonstrate that the proposed framework using the optical flow features can effectively predict control signals from a dash camera video. Using the Cityscapes dataset, we validate that the system prediction has errors as low as 0.0130 rad/s on steer angle and 0.0615 m/s on speed, outperforming existing research.

1 Introduction

Self-driving vehicles have attracted much attention due to the prospect of driving robustly without human involvement in the controls. To achieve this, the ability to analyze the environment using data coming from sensors and the ability to control the car accordingly are essential [1]. Significant effort has been dedicated to the design of vision-based autonomous driving systems thanks to the recent advancement in computer vision. In general, such research can be divided into two categories. The first way is to deploy geometric analysis algorithms to create a model of the vehicle's surrounding environment, which is then used for planning and control [2]. The second way is to learn a policy that directly maps the sensory input, such as the live feed from a front-facing camera, to a set of control commands by imitating an expert driver [3]. We follow the latter as it allows the system to directly extract features from the input that are relevant to the control problem, thereby facilitating suitable features to be extracted.

In order to mimic expert drivers, imitation learning has been applied to a variety of driving tasks, including articulated motion [4], road following [3, 5] and obstacle avoidance [6, 7]. The control policy is typically trained using supervised learning, where expert reference is provided. The system performance is improved by automatically learning the necessary internal representation from raw sensory input. Compared with rule-based solutions [2, 8], imitation learning systems are capable of recognizing visual cues that might be difficult to anticipate, allowing it to understand features that could have been missed by human drivers.

The recent development of deep learning has significantly improved the control quality in imitation learning. Bojarski et al. [3] utilize synthetic images to train an end-to-end system, which deduces the shift and rotation required for a scenario according to the recorded control signals. Toromanoff et al. [5] use a lateral control model to generate the failure cases of driving, such that machine learning systems can learn how to recover from failure cases. More recent works leverage auxiliary tasks for feature learning. Tasks such as segmentation [9] and optical flow estimation [10] can be applied to enhance the quality of training for the vehicle control prediction problem. Furthermore, multi-task frameworks have been shown to be effective in controlling multiple aspects of driving at the same time. Chowdhur et al. [6] train the system to predict the steering angle and speed simultaneously using the multi-task learning framework. They collect recovery data in real sites as a special driving behavior. Yang et al. [11] also trained a multi-task network to regress control commands and use the inferred speed to compute the recovering label. In particular, temporal information has been taking into consideration for further improving the robustness in autonomous driving. Some researchers proposed using Long Short-Term Memory (LSTM) in their deep learning frameworks to infer steering angles [9, 10, 12]. However, while these methods work well under normal circumstances, relying on a period of temporal information for driving controls could be risky as unexpected discrete events such as a car crash can happen instantaneously. How best to apply temporal information remains an open problem.

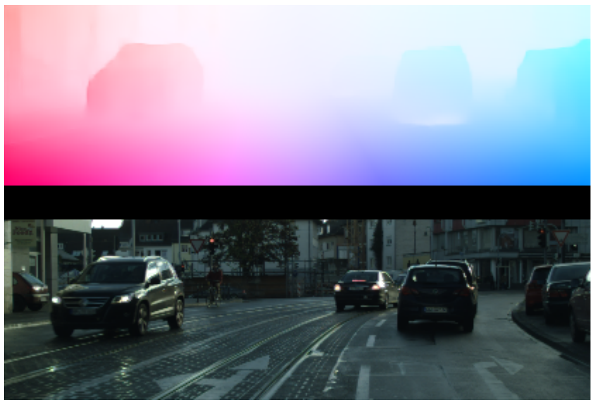


Fig. 1: ***The optical flow feature as a better cue for learning control signals*** *We proposed to generate the optical flow feature (upper) that highlights movement across frames from a color dash camera frame (lower). Such a feature is a powerful cue in learning self-driving car controls.*

In this paper, we propose a new framework for controlling self-driving cars utilizing *optical flow* [13] as a core feature to represent motion cue, as shown in Fig. 1. Such features are powerful for control tasks, as they encode the per-pixel movement between two consecutive images. This allows an effective mapping between the features and the control signals. Building around the optical flow feature, we propose a deep learning system to generate the control signals. Because the flow can encode the temporal information between two frames, our system does not need a recurrent module [9, 10, 12], although we can include such modules for considering a longer temporal duration. The frame-based nature of our system allows the system to react quickly to unexpected events.

Our framework has two major components. The first component is the flow predictor. As opposed to the approach in [6] and [9], our model does not require expensive annotations for the flow prediction task. This is made possible by a self-supervised mechanism [14] to model the underlying scene structure observed from consecutive frames and to generate the optical flow. The second component is the controller, which is a supervised convolution neural network. Its task is to learn the mapping between the input flow and the control signals. Motivated by previous success of previous work [6, 11], we design the controllers as a multi-task deep learning structure to simultaneously estimate the control speed and yaw rate. Our multi-task setup generates slightly superior results than the single-task one. It is also able to generate both control signals within the same architecture, which ensures that the two signals agree with each other to form a coordinated control strategy.

To evaluated the proposed self-driving model, the Cityscapes dataset [15] is used. The results demonstrate the effectiveness of the use of optical flow in self-driving car controls. We outperform Bojarski et al.'s [3] steer prediction network by 48.9% in Mean Absolute Error (MAE). Our multi-task network also demonstrates improvements over traditional single-task setups, boosting the MAE of steer prediction by 5.8%.

The main contributions of the paper are:

(1) We propose a new framework that utilizes optical flow as an motion cue for self-driving car controls. We demonstrate that the feature is effective in improving the accuracy of the control signal, and our deep learning system outperforms existing methods significantly.

(2) We propose a multi-task deep neural network to predict both the yaw rate and speed from the optical flow input. We show that the multi-task setup not only performs slightly better than the traditional single-task one, but also capable of generating both controls under the same architecture, ensuring the coherence of the control strategy.

The rest of the paper is arranged as follows. We first review previous work in Section 2. We explain our flow-feature informed framework including a flow predictor and a controller in Section 3. We present the Cityscapes dataset used in our experiment in Section 4 and show the results of our proposed systems in Section 5. Finally, we conclude the paper and discuss on possible future directions in Section 6.

2 Related Work

In this section, we review some prior research that is closely related to our work. We first cover learning based methods used for vehicle steering control, as well as the data required for training the system. We then discuss the extraction methods of optical flow features particularly on self-driving applications.

2.1 Deep Learning for Self-driving

Recently, as a promising approach, there is a growing interest in imitation learning to train self-driving systems. Bojarski et al. [3] proposed a deep learning framework with convolutional neural networks (CNNs) that parsed high-dimensional camera inputs through some convolutional layers into the driver's actions. The proposed system learns the internal representations of useful visual cues, such as road signs and objects, from the input images. It then correlates them with a simple control signal (i.e. the steering angle) during the training stage. This allows training for the driving policy effectively. Codevilla et al. [7] proposed a CNN framework that exploited the expert's intention representations in addition to front images for predicting the vehicle speed and steering angle. The proposed conditional imitation learning framework takes a high-level command together with the corresponding front image as the input to guide the vehicle, allowing controls such as deciding the turning direction in an upcoming intersection. Similarly, Chowdhuri et al. [6] used mode indicators as the second input in their CNN training. Specifically, three behavioral modes, namely *Direct*, *Follow* and *Furtive*, were used in the training the networks as a multi-modal multi-task learning problem. Such a secondary high-level input command enables the change of driving behavior. It also provides the system with more flexibility and can potentially adapt better to different situations. We also use CNN in our work to extract information from the optical flow feature and the camera image.

Xu et al. [9] developed a framework that integrates CNN with long short-term memory (LSTM). It learns jointly from control loss and image segmentation loss to predict control commands based on a sequence of camera frames. The main difference between their work and the prior ones is the use of uncalibrated, crowd-sourced video data in the training stage. While this makes the model training more challenging, the trained models are more robust in different situations. They further contribute to the community by making their large-scale annotated video dataset, the Berkeley DeepDrive Video Dataset (BDD-V), available to the public. We use optical flow to represent the per-pixel movement between two frames, and we achieve good results without the use of LSTM, although our design is compatible with it.

To provide the passengers and practitioners with more insight of the learned driving policies, Kim et al. [12] implemented a CNN with attention model and exploited visual explanations to interpret the steering angle predictions. In particular, the regions on the image that are influencing the driving control signals are highlighted. Recently, they [16] presented a new framework to generate textual explanations for their self-driving systems. On top of applying the attention-based models to understand the relationships between visual cues and driving control commands, the system also provides textual description for a corresponding video. Bojarski et al. [17] also proposed a method to visualize the intermediate layers to explain their CNN framework PilotNet in [3]. The key idea is to find the salient objects that are the most influential to the driving commands in the road image. Due to the simplicity of the method compared with existing saliency detection approaches such as sensitivity-based and deconvolution-based approaches, the system is fast enough to be used on the NVIDIA DRIVE^{TM} PX 2 AI car computer.

2.2 Data for Training

Training data has been a common problem for different learning techniques. Several large-scale datasets are developed to enable supervised learning in autonomous driving. Some aforementioned approaches, such as [9] and [12], were tested on public datasets that include dash camera videos with time-synchronized speed and steering commands recorded from human drivers, including the BDD100k [18] and the comma.ai datasets [19]. The KITTI dataset [20], which is the main benchmark for autonomous driving, has the front image sequences along with the camera pose. The Cityscapes dataset [15] is another benchmark. It contains preceding video frames and the time-synchronized speed (m/s) and yaw rate (rad/s). Combining them allows the calculation of the vehicle trajectory based on the kinematic model in [21]. In this research, we train our model to recognize the change in vehicle's movement from flow features such as rapid acceleration or sharp turns. Therefore, the Cityscapes dataset [15] was used as it comes with the time reference.

Unbalanced training data has introduced risks of generating unexpected control behaviours [22]. The data is captured from human drivers, and in most of cases, they are driving on a desired trajectory. This results in a dominate amount of *normal behaviors* data, and the systems have little chance to learn about recovering the vehicle from abnormal situations. Furthermore, in autonomous driving, latent shifts accumulate over time leading to the vehicle driving away from the expert trajectory [6]. The accumulated prediction error in vehicle steering angle eventually causes the vehicle to drift off the road. To tackle the problem of data bias and error accumulation, synthetic images corresponding to lateral failure cases were created to train the neural networks in [3, 5]. Reinforcement learning can be used to reduce the amount of training data needed, which rely on trial-and-error instead of pre-recorded data for training. The approaches have been used to approximate the observed values, for state or goal [23]. Still, the nature of reinforcement learning is infeasible for real-world driving due to the potential danger to road users. As a solution, Pan et al. [24] trained a driving policy by reinforcement learning in a virtual environment and adapted it to the real world. Since there was a large difference between the real-world road image and the virtual-world synthesized image, a generative adversarial network (GAN) based image translation network was proposed for image conversion. In this work, we do not focus on the data imbalance problem, but our framework is compatible with GAN for tackling it.

In autonomous driving, the visual feedback is obtained in a sparse manner and an abstracted representing the knowledge is key to an effective reinforcement learning process. To tackle this problem, Kulkarni et al. [25] combined hierarchical approaches with deep reinforcement learning to control agents. Agent behaviors and goals were simultaneously learned from experience with two levels of hierarchy. For compound learning, multiple inputs were often fused into a shared representation through concatenation [26]. This shared representation was then learned to infer the expected goals. We also apply data concatenation to combine the use of the captured color image and its corresponding optical flow for generating the control signals.

2.3 Optical Flow

The essential cues for self-driving in dynamic scenes, such as physical scene constraints, spatial patterns and existence of objects, can be obtained based on relative motion between the camera and the scene. Optical flow serves as an effective representation to estimate such information [13]. It represents the pattern of object movement over time, mostly considering two consecutive image frames [27]. It is more challenging to adapt it to outdoor environments, where the scene contains more local ambiguities resulted from reflective surfaces and less-textured surfaces [28]. This is because local features are fundamental factors for matching pixel patches when estimating the flow. One possible solution is to make use of the depth information on continuous objects which provides strong geometric cues for matching pixels in neighboring frames [29].

Optical flow has been used heavily in a wide variety of applications particularly in self-driving [30]. Kashyap et al. estimated optical flow from a front-view camera. They separated the flow into the image velocity component and the object motion using the camera motion estimated from a deep network [31]. Hou et al. [10] proposed to utilize two pre-trained auxiliary networks, one for image segmentation and one for optical flow prediction, to guide the main encoder network to generate low-dimensional deep feature, which is then inputted to an LSTM module for predicting the control signals. We believe that the use of the LSTM module to model temporal information may not be necessary, as the optical flow features already define the movement feature across time effectively. We therefore demonstrate that by using optical flow directly as the input, control signals can be accurately predicted. We further propose a multi-task structure that utilizes two sub-branches to predict two different control signals, the steering angle and speed, as opposed to [10] that outputs them in the same layer. Due to the use of individual subbranches, our system has a better capacity to generalize to control signals in which angle and speed are in some unusual combinations.

To further improve the performance of autonomous driving, researchers have been trying to estimate the 3D location of the vehicle from the ego-centric video captured from the dash camera. This can be done by estimating the camera pose (including both the location and orientation) and the spatial information between the vehicle and the environment. Such information are taken into account when learning the driving policies. Zhou et al. [14] used an explainability mask to exclude moving objects from dashcam videos. They created an unsupervised framework to simultaneously estimate the depth map and the camera egomotion with two CNNs. Remarkably, they achieved better camera pose estimation results compared with the established SLAM system. We utilize [14] to estimate the depth map and egomotion, which are then inputted to an analytical algorithm to calculate the optical flow. By modeling the scene and the dynamic objects, Casser et al. [32] improved Zhou's work by incorporating a 3D object motion estimators and achieved better results, ego motion prediction very close to the ground truth odometry and proposed a refinement method applicable to fine-tune the result in an online fashion. To jointly estimate depth and camera motion, it is common to assume that the scenes are mostly rigid. Ranjan et al. [33] used a competitive collaboration framework to reason about every pixel across the image (i.e. static or not), by exploiting low-level information like depth, camera motion and optical flow. In this research, we train a CNN controller to distinguish which of the pixel displacement is dominated by the camera motion.

3 The Optical Flow Informed Framework

We propose a deep neural network framework that consists of a flow predictor and a driving controller. The former is to construct the optical flow features, while the latter use such features to deduce driving control signals.

The overview of our framework is shown in Fig. 2. Given two consecutive frames I_{t-1} and I_t from the dash camera, the flow predictor estimates the depth images D_t at frame t and the egomotion (i.e. camera motion) $E_{t-1\rightarrow t}$ from frame $t-1$ to frame t, based on two jointly trained self-supervised convolutional neural networks (CNNs). The depth image and the egomotion are used to calculate the optical flow image $flow_{t-1\rightarrow t}$ from frame $t-1$ to frame t. The flow image is spatially concatenated with the current frame I_t to form the input of the controller, which is a supervised multi-task

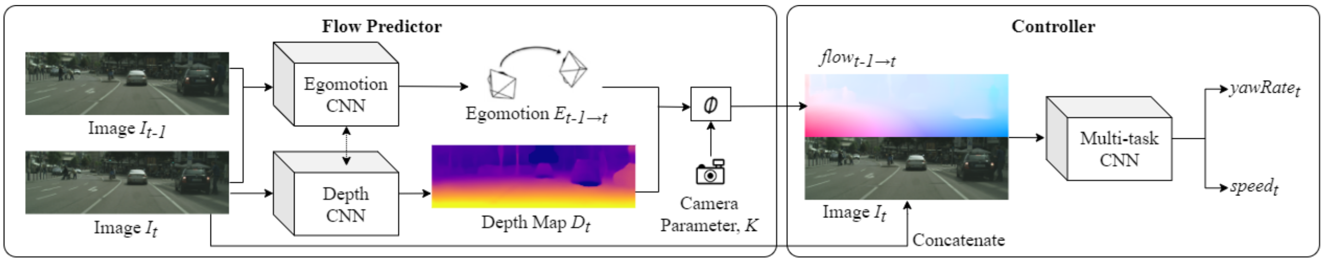


Fig. 2: ***The overview of our optical flow informed self-driving framework. Our framework consists of a flow predictor and a driving controller. The flow predictor estimates the depth image and the egomotion (i.e. camera motion) from consecutive monocular dash camera frames to calculate the optical flow. The flow and the current camera image are strong features representing the current scene. They are concatenated as the input of the driving controller, which deduces the control signals yaw rate and speed.***

CNN to predict the driving control commands, including the yaw rate $yawRate_t$ and the speed $speed_t$.

There are two major design concepts in our framework. The first one is that the front-view images and the motion-based cues between consecutive frames (i.e. optical flows) serve as a powerful representation of the current scene. Comparing with methods that do not explicitly represent the motion cues [3, 5, 12], our system performs more robustly, with the control signals predicted more accurately. The second key design is to directly use the image-based input generated from the monocular dash camera to train the control policy. Unlike existing work that apply prior knowledge to model the scene [9, 11], our framework ensures that the control policy is informed by all features obtainable from the image and flow with the least amount of user intervention.

3.1 The Flow Predictor

The flow predictor generates the optical flow image from two consecutive frames I_t and I_{t-1}, as shown in the left part of Fig. 2. To do so, we first utilize the method proposed in [14] to estimate the camera egomotion and depth map from the two frames. We then introduce an analytical algorithm to calculate the optical flow from the estimated egomotion and depth map.

Here, the dash camera should be setup to obtain a clear view with little occlusion, so that the camera motion can be effectively estimated from image features such as the geometry of the streets. An unoccluded scene is also a pre-condition for human driving, and the vast majority of available datasets fulfill this condition.

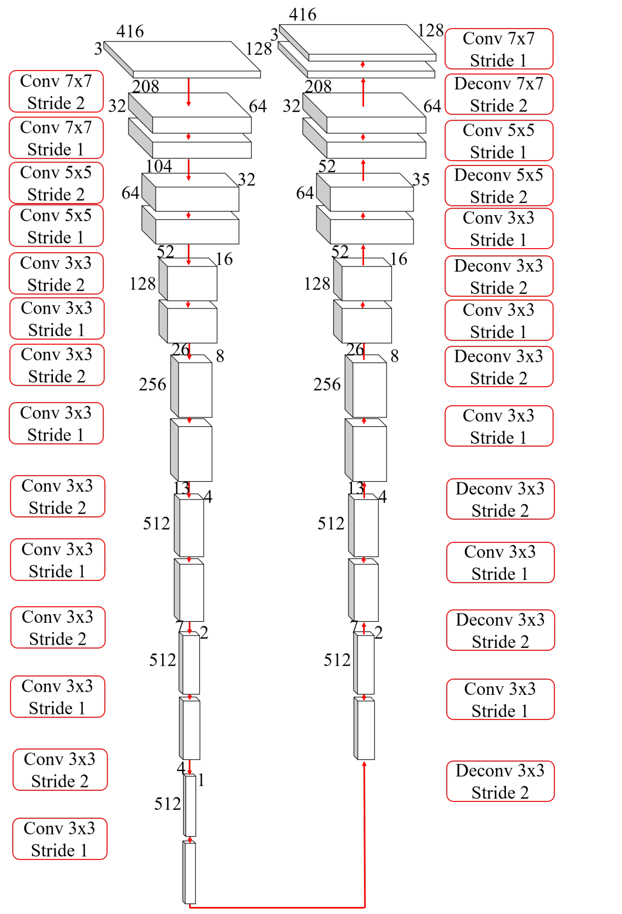


Fig. 3: ***The depth CNN for estimating the depth map*** *The depth CNN has an encoder-decoder architecture, with the input layer (top left block) being the color image and the output layer (top right block) being the estimated depth image.*

3.1.1 Self-supervised CNNs for Egomotion and Depth:

The flow predictor consists of two deep convolutional neural networks for two different tasks.

First, the depth CNN is a convolutional encoder-decoder architecture producing a dense depth map from a single RGB image. That is, it takes the frame I_t and estimates the depth map D_t. As visualised in Fig. 3, the input layer from the top left block is the RGB image. A series of convolution layers shown in the left branch reduces the images to a low-dimensional deep feature (lower left block). Then, a series of deconvolution layers shown in the right branch restores the deep feature into the original dimensionality (upper right block), with the nature of the images being converted into the depth map. In this network, the convolution layers are in pairs, consisting of one layer with a stride of 2 and another with a stride of 1. The former reduces the dimensionality of the feature, while the latter improves the generalization power of the network.

Second, the egomotion CNN takes a sequence of two RGB frames as the input and produces a 6 degree-of-freedom transformation vector between the frames as the output, represented as 3D translation and 3D rotation. That is, it takes the frames I_t and I_{t-1} to estimate the egomotion, which is the camera motion, $E_{t-1\to t}$. As visualised Fig. 4, the input layer is the image formed by concatenating I_t and I_{t-1} (top left block). The left branch is an encoder-decoder structure that reduces the image into a low dimensional deep feature (middle left block). Such a feature is passed into the right branch with future convolution layers to regress the 6 DoF camera egomotion vector.

The two networks are trained together such that the overall system is self-supervised. We implement a differentiable image warping operator ψ that reconstructs a target image from a source image with a camera motion. In each epoch, using ψ, the inverse of the egomotion $E_{t-1\to t}$ is applied on the RGB image I_t with geometry information from the depth image D_t to recreate the projected version of frame I_{t-1}, which we called $I_{t\to t-1}$. Such a projected frame is compared with the original image I_{t-1} using a photometric loss in the RGB space to construct the self-supervise signal:

$$Loss = ||I_{t\to t-1} - I_{t-1}|| \tag{1}$$

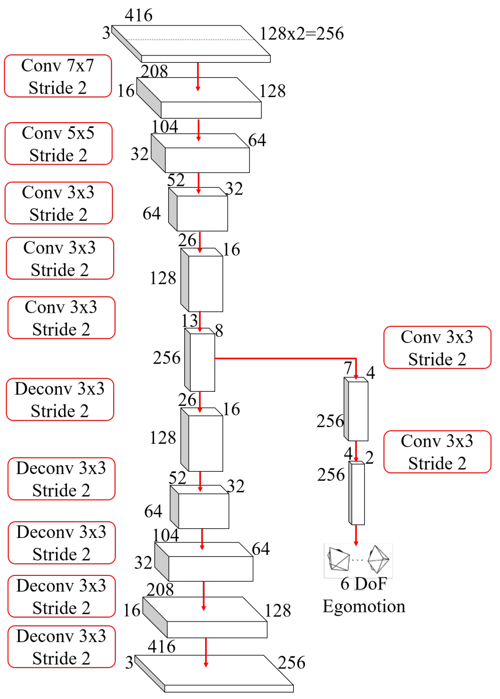


Fig. 4: ***The egomotion CNN for estimating the camera egomotion*** *The egomotion CNN has an encoder-decoder architecture as shown in the left branch, with the input layer (top left block) being the two concatenated color images and the output layer (lower right block) being the estimated 6 DoF camera egomotion.*

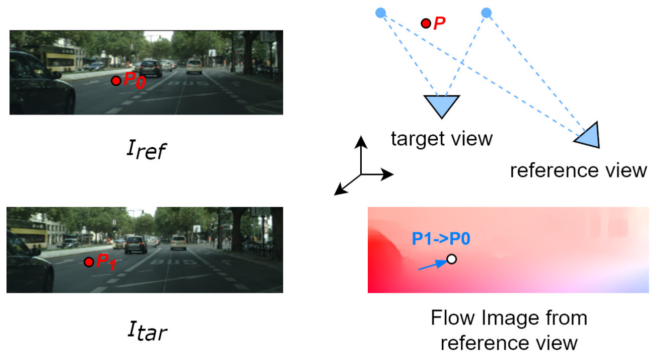


Fig. 5: ***The geometric relationship between reference and target views.*** *Given a car is turning right and merging into a traffic lane. Point P is a static point on the road surface. The pixel point P_0 is the point P from a reference view I_{ref}. The pixel point P_1 is the point P from a target view I_{tar}. The flow image is from the reference view I_{ref}.*

The loss informs the performance of both the egomotion CNN and the depth CNN, and is used to train both systems. Regarding the implementation, we scale the images in our dataset to fit the input dimensions of the depth and pose networks, which helps to avoid changing the network structure.

3.1.2 Optical Flow Calculation:

A major requirement for self-driving systems is to understand highly dynamic environments and extract robust features. This motivates us to represent the motion features using dense optical flow, which indicates the displacement of all pixels between two consecutive frames. On the one hand, traditional color images, depth images segmentation features used in self-driving vehicles [14] [36] can only represent a static frame. Therefore, they typically require the deep learning network to include a temporal module such as LSTM [10] for understanding motion and predict the control signals. With the optical flow that represents motion features directly, we demonstrate that the temporal module is not necessary for our system. On the other hand, traditional features for temporal tracking such as HOG and HOF are 2D in nature, and the tracking performance depends heavily on whether distinctive feature points are available [37]. Our optical flow is calculated by first projecting the 3D point cloud from the depth image with the camera egomotion, and then mapping the angle and magnitude of the per-pixel 3D motion into the 2D color space, as shown in Fig. 1 as an example. Therefore, our optical flow represents dense 3D motion features for self-driving controls.

As illustrated in Fig. 5, when an observed point P on a surface remains static between the two frames, the observed optical flow is purely driven by the camera motion. For the sake of better understanding, an example of a car turning right and merging into a traffic lane is used here. Point P is a static point on the road surface. The pixel point P_0 is the point P from a reference view I_{ref}. The pixel point P_1 is the point P from a targeted view I_{tar}. Since the camera moves along with the car, the pixel point P_0 moves to P_1 accordingly.

More specifically, we calculate the optical flow field due to the camera motion. Given two consecutive images I_{t-1} and I_t, we first estimate the depth map D_t and the relative camera pose $E_{t-1\to t}$ by the self-supervised CNNs and Eq. 2 mentioned in Section 3.1.1. We then calculate the optical flow:

$$flow_{t-1\to t} = \phi(D_t, E_{t-1\to t}, K) \tag{2}$$

where ϕ is an analytical algorithm, K is a matrix representing the camera's intrinsic parameters.

The algorithm ϕ can be broken down into multiple key steps. First, we create two matrices u and v that represent pixel displacement along the x and y axes respectively:

$$u = \begin{bmatrix} 0 & 1 & ... & w \\ 0 & 1 & ... & w \\ & & \ddots & \\ 0 & 1 & ... & w \end{bmatrix}, v = \begin{bmatrix} 0 & 0 & ... & 0 \\ 1 & 1 & ... & 1 \\ & & \ddots & \\ h & h & ... & h \end{bmatrix} \tag{3}$$

where w and h are the width and height of the input image I_{t-1} and I_t. We then use u and v to represent the depth map D_t in a point cloud representation consisting of the x, y and z components:

$$PC = [PC_x, PC_y, PC_z, 1] \tag{4}$$

where

$$\begin{aligned} PC_x &= D_t \odot u \\ PC_y &= D_t \odot v \\ PC_z &= D_t \end{aligned}$$

where $\odot$ represent the element-wise multiplication. On the other hand, we estimate the normalized camera egomotion considering the intrinsic parameters K:

$$E_{t-\hat{1}\to t} = K \cdot E_{t-1\to t} \cdot K^{-1} \tag{5}$$

where K^{-1} denotes the inverse of K. We then project the point cloud with the normalized egomotion to obtain the movement matrix:

$$\hat{PC} = E_{t-\hat{1}\to t} \cdot PC^T \tag{6}$$

where PC^T denotes the transpose of the PC matrix. Finally, the two components of the optical flow are calculated as:

$$flow_{t-1\to t,u} = \frac{\hat{PC_x}}{\hat{PC_z}} - u \tag{7}$$

$$flow_{t-1\to t,v} = \frac{\hat{PC_y}}{\hat{PC_z}} - v \tag{8}$$

where PC_x, PC_y and PC_z are the x, y and z component of PC respectively. The 2D flow image $flow_{t-1\to t}$ is represented by mapping the angle and magnitude of the optical flow into the color space.

The flow $flow_{t-1\to t}$ and the image I_t are considered as strong features and are used as the input of the CNN-based controller network.

In addition, following [14], we implement a depth smoothing strategy. Specifically, we trained our flow predictor on the KITTI dataset [20] and then applied the online refinement technique proposed in [32] to fine-tune the flow prediction module in this work. We found that the predicted flow fields successfully reflect the scene level consistent movement governed by the camera motion.

3.2 The Controller Network

In this section, we explain the controller network that predicts the control signals yaw rate and speed through a multi-task network architecture. Such an approach improves the generalization and exploiting commonalities between the two tasks in order to achieve a better performance in controlling the vehicle.

We adapt the end-to-end CNN architecture in [3], as its effectiveness in mapping raw pixels directly to vehicle control signals has been demonstrated. The ene-to-end architecture means that images features are mined directly from the raw input with minimal human interventions.

We modify the architecture in [3] to create our controller network by introducing a multi-modal multi-task CNN structure, as shown in

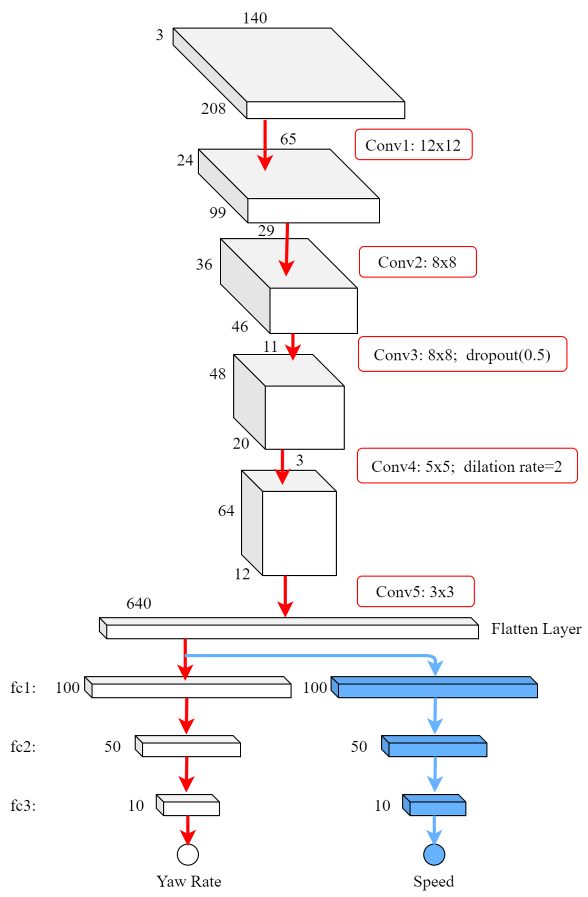


Fig. 6: ***The CNN architecture for our controller module.*** *The network takes multi-modal input (i.e. image and optical flow) and performs multitask control prediction (i.e. yaw rate and speed).*

Fig. 6. From the top of the image, the network takes the concatenated image of the current image, I_t, and the optical flow, $flow_{t-1\to t}$ as the input, which is a 3-channel RGB image. The 5 convolution layers extract the image features that are useful to our task. The flatten layer generate a 1-D vector that represents the deep features. Then, the left hand side part of the network colored in white, consisted of 3 fully connected layers, generates the yaw rate, $yawRate_{t\to t+1}$. The right hand side part colored in blue, consisted of 3 fully connected layers as well, generates the speed, $speed_{t\to t+1}$.

This multi-task structure allows the system to model the correlation between different tasks, which are the steering angle and speed in our case. The shared layers are used to model common knowledge between the two tasks, while the separate sub-branches are used to model the task-specific knowledge. During backpropagation, the ground-truth signal of each task update both the shared layers and its task-specific branch. Through the shared layers, the knowledge between the two tasks is generalized.

The loss function of our framework is defined as the mean absolute error between the predicted yaw rate $yawRate_{t\to t+1}$ and the ground truth $yawRate'_{t\to t+1}$, together with that between the predicted speed $speed_{t\to t+1}$ and the corresponding ground truth $speed'_{t\to t+1}$:

$$Loss = \alpha_1 ||yawRate_{t\to t+1} - yawRate'_{t\to t+1}|| + \alpha_2 ||speed_{t\to t+1} - speed'_{t\to t+1}|| \quad (9)$$

where α_1 and α_2 are the parameters to balance the influence of yaw rate and speed loss respectively on the prediction task. An ablation study with different loss weight ratios α_1/α_2 has been conducted to find out the best prediction performance, which is described in Section 5.

Furthermore, in terms of implementation, different from the CNN architecture in [3], we adopt a larger kernel size of 12×12 in the first few convolutional layers. This is driven by the observation that larger kernels are more suitable for environmental contexts extraction from front-view cameras [11]. Bojarski et al. [3] resizes the input to 66 × 200 pixels, which limits the network's ability to learn from smaller objects in the large scene. Motivated by [34] where contextual representations are learned from a large receptive field using dilated convolution layers, we increase the dilation rate in the convolutional operation, which allows learning from the images with less computational cost with a larger dimensions of 140 × 208 pixels.

3.2.1 The Baseline Setup: We set up a baseline network with only the yaw rate network but taking the multi-modal input of an image frame and the corresponding optical flow. This allows us to demonstrate the effectiveless of our multi-task architecture that considers both yaw rate and speed.

The baseline network is represented as the white part of the network in Fig. 6 without the blue part. The network still takes the multi-modal input of both the current image, I_t, and the optical flow, $flow_{t-1\to t}$, but it only predicts the yaw rate, $yawRate_{t\to t+1}$.

The loss function is defined as:

$$Loss = ||yawRate_{t\to t+1} - yawRate'_{t\to t+1}|| \quad (10)$$

where $yawRate'_{t\to t+1}$ is the corresponding ground-truth value.

4 The Cityscapes Dataset

In this section, we introduce the benchmark Cityscapes dataset [15] and the data pre-processing methods used in this work. Such a dataset fits the best in our system for several reasons. First, Cityscapes provides both the vehicular ego motion data and the camera intrinsic parameters, which are both necessary information in our system. Other popular datasets such as BDD100K [18] do not provide camera information. Second, it provides control information including both speed and steering angle, which is not present in other datasets such as KITTI [20]. Third, the dataset consists of many rapid acceleration/deceleration periods and sharp turns in a complicated city environment, which serve as challenging scenarios for the system to learn the driving signal. In comparison, the comma.ai [19] dataset consists of mainly highway scenarios, and the Ford Campus Vision and Lidar Data Set [35] consists of mainly straight roads. Both are considered to be less challenging.

The Cityscapes dataset is a large-scale dataset with mainly urban street scenes. It contains 5,000 preceding video sequences and time-synchronized ego-motion parameters from the vehicle odometry including the speed (m/s) and the yaw rate (rad/s) annotation for every frame. Every video sequence has 30 frames collected within 1.8 seconds. Because images sampled at a high rate would be highly similar and thus provide less useful information [3], we extracted pairs of images and ego-motion labels from each video sequence at the same sampling rate as in [32] for feature extraction. Furthermore, unnecessary parts of the images such as the automobile hood and the sky were cropped out to improve the efficiency of the training process. The resultant images are scaled for normalization to match the input size of the networks.

Video sequences in the Cityscapes dataset were collected in the daytime under good or medium weather conditions in 50 cities, with 4 types of camera intrinsic settings. Some examples of the contained street scenes with varying background and scene layout are shown in Fig. 7 and Fig. 9.

4.1 Annotating data using the speed ratio

In order to make the control predictions smooth and conform to motion laws, we tried to adopt the speed ratio that compares the speed of the current frame with the previous frame as the label as in [11] and [16]. We would like our controller to regress a vehicle's movement trend $\Delta s = s_i/s_{i-1}$ from the RGB image and the flow image, and then determine the speed s_i based on the vehicle's prior movement state s_{i-1}. For both of the aforementioned speed label preparation methods, the samples with a very low speed (less than 0.1 m/s) are discarded to ensure a consistent driving condition. We demonstrate the performance gain and justify our choice in using speed ratio in an ablation study in Section 5.4.1.

5 Experimental Results

In this section, both quantitative and qualitative experimental results are presented. In particular, we compare the proposed multi-task network with [3] and the baseline model explained in Section 3.2.1, demonstrating that our multi-task system performs the best.

5.1 *Experiments Setup*

We used the same preprocessing settings and neural network hyperparameters for all the models. Samples were partitioned according to the setup of the Cityscapes dataset (59.5% training, 10% validation and 30.5% testing) and were randomly shuffled. All the models were trained and tested using Keras utilizing a Tensorflow backend on a GeForce GTX 1080 GPU based system. We used the Adam optimizer with $\beta_1 = 0.99$, $\beta_2 = 0.999$ and a fixed learning rate of 1×10^{-4} in all experiments. The training time for the multi-task network was about 5 hours for 30 epochs.

5.2 *Qualitative Evaluation*

Some examples of video frames and the corresponding intermediate images (i.e. depth maps and flow features) are illustrated in Fig. 7 to Fig. 10 to demonstrate the high-quality results obtained using our proposed framework. In particular, brighter color in the depth map denotes greater relative motion and shorter distance. Brighter color in the optical flow features indicates a larger drift or movement. It can be seen that the extracted features are highly indicative of steering angle prediction (the going through and turning behaviors) in Fig. 9 and Fig. 10. In addition, the depth information on objects like cars, pedestrians and bicycles provides strong geometric cues for inference, which can be seen in Fig. 7 and Fig. 8.

The depth maps computed from a wide range of different situations are presented in Fig. 7 and Fig. 8. The samples in Fig. 7 (a), (b) and (c) clearly highlights the spatial distances between the camera and other vehicles are estimated accurately. In Fig. 8 (a), it can be seen that the shapes and depth information of the objects (i.e. cars and pedestrian) is well-captured. We further show 2 challenging scenes in Fig. 8 (b) and (c) in which a cyclist is included in the scenes with moderate traffic. Again, our proposed method extracted high-quality shape and depth information.

Examples of the flow features computed by our method can be found in Fig. 9 and Fig. 10. The results indicate that our method produces smooth and accurate flow features in different situations consistently. When the cars are moving together in relatively high speed, the flow features indicates that correctly in Fig. 10 (b) and (c). The samples in Fig. 9 (a), (b) and (c) do not contain a lot of moving vehicles. As a result, the region surrounding the vanishing point will have larger changes and indicated by a brighter color in the flow features. Fig. 10 (a) shows a traffic jam scenario with vehicles move slowly. Similar to Fig. 9 (a) to (c), the region surrounding the vanishing point will have larger movement.

5.3 *Quantitative Evaluation*

5.3.1 *System Performance:*

The performance of the proposed multi-task model to predict both yaw rate and speed on the Cityscapes dataset is shown in Table 1. We show the mean average error (MAE) and root mean square error (RMSE) for the yaw rate and the speed ratio.

It can be noted that the minimum yaw rate error is obtained with the weight ratio $\alpha_1 : \alpha_2 = 1 : 1$, in which the MAE is 0.0130 and RMSE is 0.0281. The minimum speed error is obtained with the weight ratio $\alpha_1 : \alpha_2 = 1 : 3$, in which the MAE is 0.0615 and the RMSE is 0.0867.

It can also be observed that the weight ratio does not significantly increase the error, making it easy to pick the parameters that does well on both yaw rate and speed. For example, the parameter ratio $2 : 1, 1 : 1, 1 : 2$ and $1 : 3$ all generate good results in both yaw rate and speed, and they all outperform other methods we compare with. Such comparisons will be explained in the next section.

5.3.2 *Comparison with Other Methods:*

We compare the proposed multi-task system with [3] and the baseline model as explained in Section 3.2.1 on the Cityscapes dataset. Since both [3] and the baseline can only generate yaw rate, we optimized the models with respect to the yaw rate.

The result is shown in Table 2. Our optimized multi-task model outperforms both [3] and the baseline, with the smallest mean average error (MAE) and the smallest root mean square error (RMSE). The multi-task model is set up using the parameter ratio $\alpha_1 : \alpha_2 = 1 : 1$ and using the speed ratio as the target of the regression.

In particular, compared with [3], we observe an improvement in MAE of yaw rate from 0.0270 rad/s to 0.0138 rad/s, which is equivalent to a relative improvement of 48.9%. The RMSE is reduced from 0.0487 rad/s to 0.0282 rad/s, by 42.1%. It highlights the effectiveness of using flow features to help the model better understand the scene geometry and spatial pattern and thus generate a much more accurate yaw rate prediction.

Also, compared with the baseline, the proposed system improved the MAE from 0.138 to 0.130 and RMSE from 0.282 to 0.281, despite the need of solving two tasks at the same time. This shows that the two tasks are highly relevant to each other. The proposed multi-task framework effectively solves them in the same network and generates better results.

While the best result for the yaw rate is obtained with the weight ratio $\alpha_1 : \alpha_2 = 1 : 1$, we can see from Table 1 that even when we pick the weight ratio that is optimized for speed (i.e. $\alpha_1 : \alpha_2 = 1 : 3$), the proposed multi-task system still performs better than [3] and the baseline.

We further compare our work with [36], which achieved the second-best performance in the ICCV 2019 Learning to Drive challenge. It comes with the source code that we can use to retrain the network with the Cityscapes dataset for experimental comparisons. We include both the single image (CNN_SINGLE_IMAGE) and stacked (CNN_STACKED) implementations in [36] for comparison. As shown in Table 3, it can be seen that aligning with the findings in [36], the stacked implementation performs better. However, both networks cannot produce results as good as our proposed system, either using the parameter ratio of 1:1 or 1:3. This indicates that the

Table 1 The yaw rate and speed prediction errors with the proposed multi-task model on Cityscapes.

$\alpha_1 : \alpha_2$	Yaw Rate MAE (rad/s)	Yaw Rate RMSE (rad/s)	Speed MAE (m/s)	Speed RMSE (m/s)
2:1	0.0132	0.0287	0.0632	0.0885
1:1	**0.0130**	**0.0281**	0.0649	0.0902
1:2	0.0132	0.0278	0.0635	0.0885
1:3	0.0135	0.0287	**0.0615**	**0.0867**
1:4	0.0139	0.0291	0.0627	0.0880
1:5	0.0146	0.0300	0.0627	0.0874
1:7	0.0148	0.0291	0.0629	0.0882

Bold values are best results.

Table 2 Comparisons with [3] and baseline on Cityscapes

Model	Yaw Rate MAE (rad/s)	Yaw Rate RMSE (rad/s)
Bojarski et al.'s model [3]	0.0270	0.0487
The baseline model (explained in Section 3.2.1)	0.0138	0.0282
The proposed multi-task model	**0.0130**	**0.0281**

Bold values are best results.

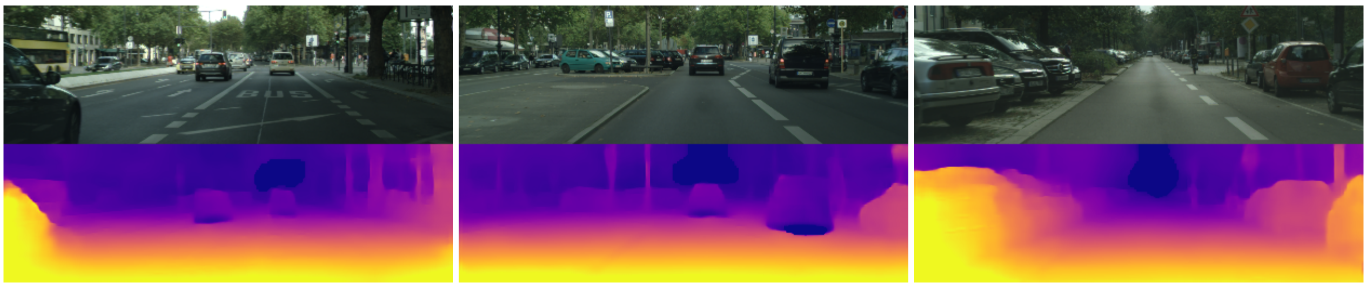

(a) Another car in proximity (b) Typical traffic (c) A lot of parked cars

Fig. 7: ***Examples of depth maps computed from different situations.***

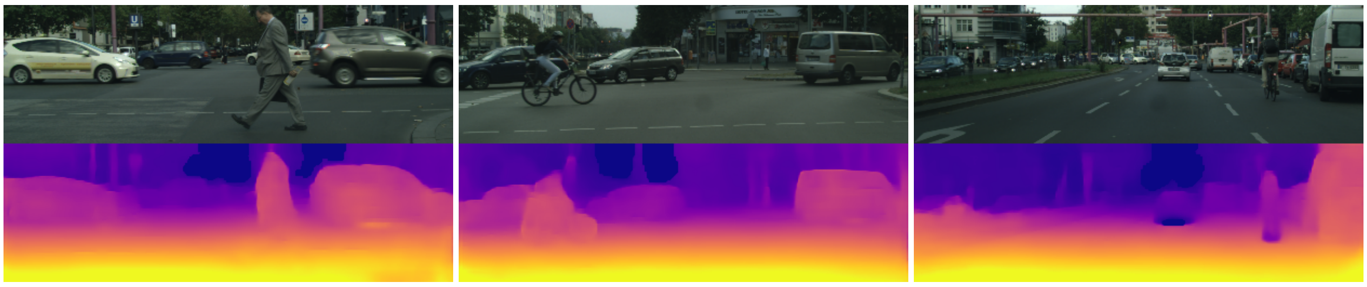

(a) With pedestrian (b) More complicated scene with a cyclist (c) A more complicated scene with a cyclist

Fig. 8: ***Examples of depth maps computed from different situations.***

use of optical flow has a positive effect on estimating the control signals.

5.4 *Ablation Studies and Parameter Analysis*

5.4.1 The Speed Label Preparation Strategy: Here, we further evaluate the performance of incorporating the multi-task model in the proposed framework. Specifically, this ablation test focuses on the effect of the strategy of using the speed ratio annotated data in the proposed multi-task model.

For yaw rate prediction, as shown in Fig. 11(a), the use of the previous dynamic state produced encouraging results by bringing down the yaw rate MAE for all the tests with different loss weight ratios. Without the use of previous dynamic state, the minimal MAE of 0.0144 was achieved when using 1:2 loss weight for yaw rate and speed prediction. With the use, the minimal MAE of 0.0130 was achieved when using 1:1 loss weight.

Similarly, for speed prediction, the use of previous dynamic state improved the results significantly in all weight ratios as shown in Fig. 11(b). Without it, the minimal MAE of 1.9177 was achieved with the weight ratio of 2:1. With it, the minimal MAE of 0.0615 was achieved with the weight ratio of 1:3.

5.4.2 The Loss Weight Ratio: Here, we present the results of an ablation study on changing the loss weight ratios in our multi-task model. The results are shown in Fig. 12. The best yaw rate prediction performance is achieved with the 1:1 loss weight ratio, while the best speed prediction is obtained with the ratio of 1:3.

Table 3 Comparisons with [36] on Cityscapes

Model	Yaw Rate MAE (rad/s)	Yaw Rate RMSE (rad/s)	Speed MAE (m/s)	Speed RMSE (m/s)
[36] Single Image	0.0470	0.0759	2.6911	3.4776
[36] Stacked	0.0246	0.0441	1.9261	2.5235
Our model (1:1)	**0.0130**	**0.0281**	0.0649	0.0902
Our model (1:3)	0.0135	0.0287	**0.0615**	**0.0867**

Bold values are best results.

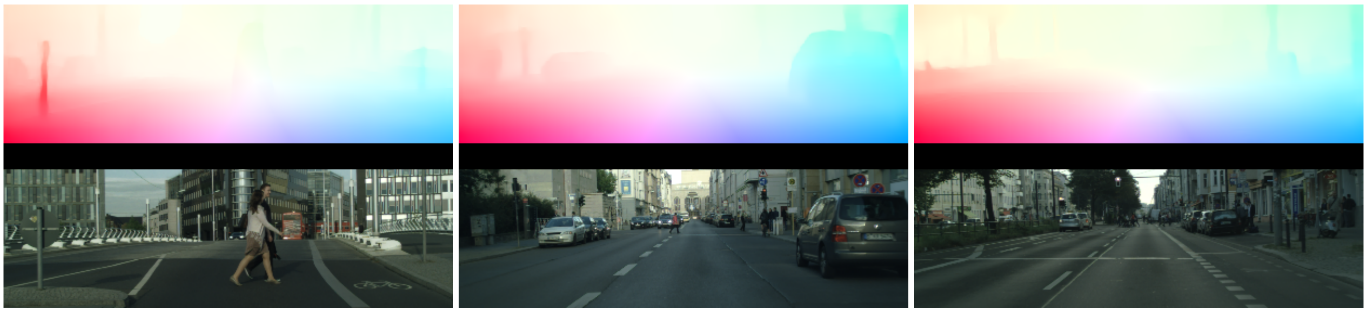
Notice that our system performs generally well in the ratio range between 2:1 and 1:4. For example, using the ratio 1:1 where yaw weight achieves the best result, the speed error is only increased by 5%. Using the ratio of 1:3 where the speed achieves the best result, the yaw rate error is only increased by 4%. Finally, the ratio 1:2 resulting in a balance between yaw rate and speed.

6 Conclusion

Without learning from motion-based cues, vehicle control prediction networks are susceptible to complicated road scenes. In this paper, we develop a CNN framework to learn from dashcam videos and flow features between frames to predict driving control commands including yaw rate and speed. Our experiments show that flow features help the proposed approach to achieve a new state-of-the-art on the Cityscapes benchmark. With the speed prediction task, our multi-task network largely outperforms the previous steer prediction.

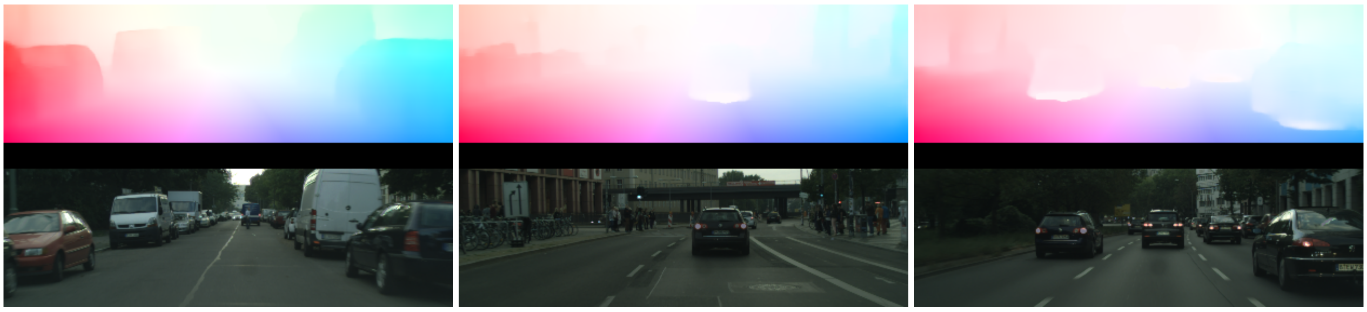
In the future, we are interested in implementing a driving test with a pre-recorded video in a simulator to further evaluate our work. Specifically, the simulator sends the first two frames of the test video to our trained model. The flow predictor creates the flow image and then the controller network infers the control commands for the second frame. The control predictions including speed and yaw rate are fed into the vehicle kinematic model [21] to compute the yaw and the displacement travelled by the virtual car. We will assume the ground is flat so we can transform the car's yaw and displacement in the world coordinate system to a camera motion in the camera coordinate system with camera calibration.

With the camera pose and multi-view images, we are able to synthesize a novel view for the updated position and orientation of the car. Ideally, the simulator then sends the new synthetic frame and the second frame to the model and the process repeats. Since the camera motion is available when creating the flow image and our model also predict control commands based on the flow image, it is convenient to calibrate the lane center associated with each frame and augment recovering data for failure case using the formula in [11]. By observing whether the simulated car would drift away in time, we will be able to evaluate the performance of the controller in managing to perform lane-keeping and car following tasks.



(a) Pedestrians crossing the road slowly (b) Low traffic (c) Low traffic

Fig. 9: ***Examples of flow features computed from different situations.***



(a) A lot of parked cars (b) Cars moving together with higher speeds (c) Cars moving together with higher speeds

Fig. 10: ***Examples of flow features computed from different situations.***

While this research can accurately predict the control signals from a video feed, control signal prediction is only one part of a fully self-driving vehicle system. When deployed onto a car, issues such as control signal delay and control noises would affect the system. With the use of driving simulation systems or even real cars, we may perform closed-loop tests to identify potential gaps in the control loop. In this aspect, we may also look into more dedicated control systems such as feedback controls to fill the gaps.

Acknowledgement

We would like to thank Zisu Xie for the assistance in generating the experimental results. The project was supported in part by the Royal Society (Ref: IES\R2\181024 and IES\R1\191147).

7 References

[1] Paden, B., Čáp, M., Yong, S., *et al.*: 'A survey of motion planning and control techniques for self-driving urban vehicles', IEEE Transactions on Intelligent Vehicles, 2016, 1, (1), pp. 33–55

[2] Chen, C., Seff, A., KornhauserJones, A., *et al.*: 'DeepDriving: Learning affordance for direct perception in autonomous driving'. IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, December 2015, pp. 2722–2730

[3] Bojarski, M., Del Testa, D., Dworakowski, D., *et al.*: 'End to end learning for self-driving cars'. arXiv preprint arXiv:1604.07316, 2016

[4] Hemachandra, S., Duvallet, F., Howard, T. M., *et al.*: 'Learning models for following natural language directions in unknown environments'. IEEE International Conference on Robotics and Automation (ICRA), Seattle, USA, May 2015, pp. 5608-5615

[5] Toromanoff, M., Wirbel, E., Wilhelm, F., *et al.*: 'End to end vehicle lateral control using a single fisheye camera'. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October 2018, pp. 3613–3619

[6] Chowdhuri, S., Pankaj T., Zipser K.: 'MultiNet: Multi-modal multi-task learning for autonomous driving'. IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa Village, USA, January 2019, pp. 1496–1504

[7] Codevilla, F., Miiller, M., López, A., *et al.*: 'End-to-end driving via conditional imitation learning'. IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, August 2018, pp. 1–9

[8] Broad, A., Arkin, J., Ratliff, N., *et al.*: 'Real-time natural language corrections for assistive robotic manipulators'. International Journal of Robotics Research, 2017, 36, (5-7), pp. 684-698

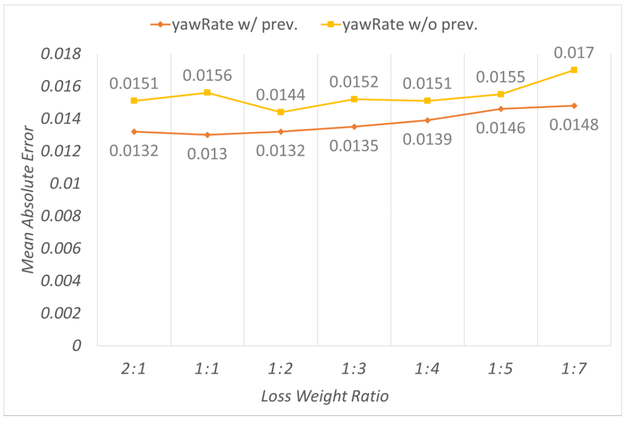
[9] Xu, H., Gao, Y., Yu, F., *et al.*: 'End-to-end learning of driving models from large-scale video datasets'. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, July 2017, pp. 3530–3538

[10] Hou, Y., Ma, Z., Liu, C., *et al.*: 'Learning to steer by mimicking features from heterogeneous auxiliary networks'. AAAI Conference on Artificial Intelligence, New Orleans, USA, February 2018, pp. 8433–8440

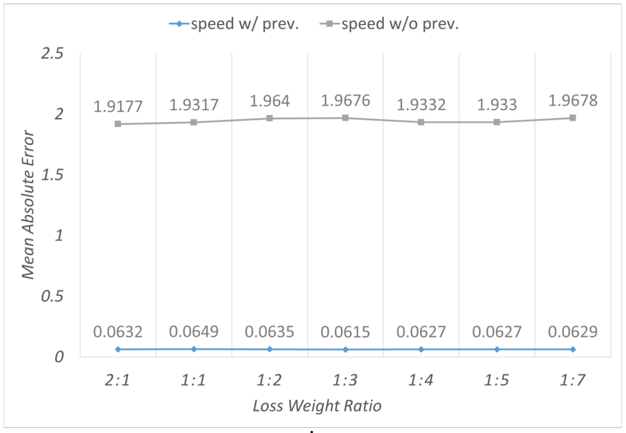
[11] Yang, Z., Zhang, Y., Yu, J., *et al.*: 'End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions'. International Conference on Pattern Recognition (ICPR), Beijing, China, August 2018, pp. 2289–2294

[12] Kim, J., Canny, J.: 'Interpretable learning for self-driving cars by visualizing causal attention'. IEEE International Conference on Computer Vision (ICCV), Venice, Italy, October 2017, pp. 2961–2969

[13] Brox, T., Bruhn, A., Papenberg, N., *et al.*.: 'High accuracy optical flow estimation based on a theory for warping'. European

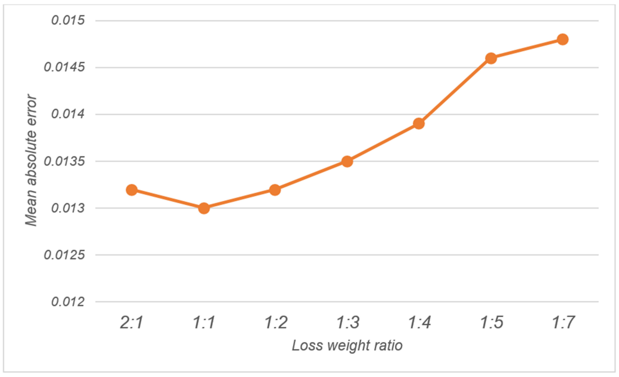


a

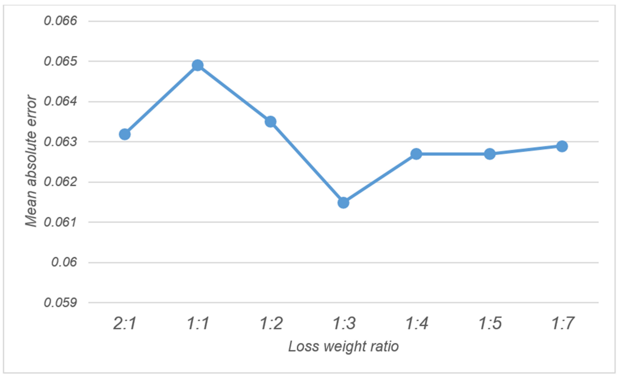


b

Fig. 11: ***Experiment results of multi-task network with/without using the speed ratio.*** (a) Yaw rate prediction of multi-task CNN with different loss weight ratios. (b) Speed prediction of multi-task CNN with different loss weight ratios.



a



b

Fig. 12: ***Experiment results of multi-task network with different loss weight ratios for yaw rate and speed.*** (a) Yaw rate prediction of multi-task CNN. (b) Speed prediction of multi-task CNN.

Conference on Computer Vision (ECCV), Prague, Czech Republic, May 2004, pp. 25–36

[14] Zhou, T., Brown, M., Snavely, N., *et al.*: ‘Unsupervised learning of depth and ego-motion from video’. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, July 2017, pp. 6612–6619

[15] Cordts, M., Omran, M., Ramos, S., *et al.*: ‘The cityscapes dataset for semantic urban scene understanding’. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, June 2016, pp. 3213–3223

[16] Kim, J., Rohrbach, A., Darrell, T., *et al.*: ‘Textual explanations for self-driving vehicles’. European Conference on Computer Vision (ECCV), Munich, Germany, September 2018, pp. 577–593

[17] Bojarski, M., Yeres, P., Choromanska, A., *et al.*: ‘Explaining how a deep neural network trained with end-to-end learning steers a car’. arXiv preprint arXiv:1704.07911, 2017

[18] Yu, F., Xian, W., Chen, Y., *et al.*: ‘BDD100K: Adiverse driving video database with scalable annotation tooling’. arXiv preprint arXiv:1805.04687, 2018

[19] Santana, E., Hotz, G.: ‘Learning a driving simulator’. arXiv preprint arXiv:1608.01230, 2016

[20] Geiger, A., Lenz, P., Urtasun, R.: ‘Are we ready for autonomous driving? The KITTI vision benchmark suite’. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, USA, June 2012, pp. 3354–3361

[21] Wang, D., Qi, F.: ‘Trajectory planning for a four-wheel-steering vehicle’. IEEE International Conference on Robotics and Automation (ICRA), Seoul, South Korea, May 2001, pp. 3320–3325

[22] Zhang, J., Cho, K.: ‘Query-efficient imitation learning for end-to-end simulated driving’. AAAI Conference on Artificial Intelligence, San Francisco, USA, February 2017, pp. 2891-2897

[23] Schaul, T., Horgan, D., Gregor, K., *et al.*: ‘Universal value function approximators’. International Conference on Machine Learning, Lille, France, July 2015, pp. 1312-1320

[24] Pan, X., You, Y., Wang, Z., *et al.*: ‘Virtual to Real Reinforcement Learning for autonomous driving’. British Machine Vision Conference, London, UK, September 2017, pp. 11.1-11.13

[25] Kulkarni, T. D., Narasimhan, K., Saeedi, A., *et al.*: ‘Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation’. Advances in Neural Information Processing Systems, Barcelona, Spain, December 2016, pp. 3682-3690

[26] Eitel, A., Springenberg, J. T., Spinello, L., *et al.*: ‘Multimodal deep learning for robust RGB-D object recognition’, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, September 2015, pp. 681-687

[27] Sun, D., Roth, S., Black, M.J.: ‘Secrets of optical flow estimation and their principles’. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, USA, June 2010, pp. 2432–2439

[28] Young, G., Hong, T., Herman, M., *et al.*: ‘Obstacle detection for a vehicle using optical flow’. Intelligent Vehicles 92 Symposium IEEE, 1992, pp. 185–190

[29] Behl, A., Jafari, O. H., Mustikovela, S. K., *et al.*: 'Bounding Boxes,segmentations and object coordinates: How important is recognition for 3D scene flow estimation in autonomous driving scenarios?', IEEE International Conference on Computer Vision (ICCV), Venice, Italy, October 2017, pp. 2593-2602

[30] Song, K., Chen, H.: 'Lateral driving assistance using optical flow and scene analysis'. IEEE Intelligent Vehicles Symposium, 2007, pp. 624–629

[31] Kashyap, H., Fowlkes, C., Krichmar, J.: 'Sparse Representations for Object and Ego-motion Estimation in Dynamic Scenes'. arXiv preprint arXiv:1903.03731, 2019

[32] Casser, V., Pirk, S., Mahjourian, R., *et al.*: 'Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos'. AAAI Conference on Artificial Intelligence, Honolulu, USA, January 2019, pp. 8001–8008

[33] Ranjan, A., Jampani, V., Balles, L., *et al.*: 'Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation'. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, USA, June 2019, pp. 12240–12249

[34] Yu, F., Koltun, V., Funkhouser, T.: 'Dilated Residual Networks'. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, July 2017, pp. 636–644

[35] Pandey, G., McBride, J. R., Eustice, R. M: 'Ford campus vision and lidar data set'. The International Journal of Robotics Research, 30(13), pp. 1543–1552, 2011

[36] Lovjer, A., Yeom, M., Schifferer, B., Drori, I., 'Using Segmentation Masks in the ICCV 2019 Learning to Drive Challenge', arXiv preprint, https://arxiv.org/abs/1910.10317, 2019

[37] Kong, X., Chen, Q., Gu, G., Ren, K., Qian, W., Liu, Z., 'Particle filter-based vehicle tracking via HOG features after image stabilisation in intelligent drive system'. IET Intelligent Transport Systems, vol. 13, no. 6, pp. 942-949, 6, 2019