2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Autonomous Robot Navigation Based on Multi-Camera Perception

Kunyan Zhu†, Wei Chen†, Wei Zhang*, Ran Song, and Yibin Li

***Abstract*— In this paper, we propose an autonomous method for robot navigation based on a multi-camera setup that takes advantage of a wide field of view. A new multi-task network is designed for handling the visual information supplied by the left, central and right cameras to find the passable area, detect the intersection and infer the steering. Based on the outputs of the network, three navigation indicators are generated and then combined with the high-level control commands extracted by the proposed MapNet, which are finally fed into the driving controller. The indicators are also used through the controller for adjusting the driving velocity, which assists the robot to adjust the speed for smoothly bypassing obstacles. Experiments in real-world environments demonstrate that our method performs well in both local obstacle avoidance and global goal-directed navigation tasks.**

I. INTRODUCTION

We have observed significant progress in various sub-tasks of autonomous robot navigation such as scene perception [1][2], simultaneous localization and mapping [3][4], path planning [5][6] and lane keeping [7][8]. However, it is still challenging to design a navigation system that integrates goal-directed navigation with obstacle avoidance in an unstructured environment, such as a campus site and a main street crowded with pedestrians and cars. This requires the robot to be able to handle diverse scenarios based on sufficient perception of its surrounding environment.

With Deep Learning demonstrating state-of-the-art performance in various vision tasks and the low cost of RGB cameras, vision-based navigation methods have received much attention. There are two major paradigms for autonomous vision-based navigation: mediated perception and data-driven behavior reflex [9]. The former is suitable for structured environment and makes decisions mainly based on the mediated representation including sub-tasks related to autonomous navigation, such as segmentation [10][11], object detection [12][13] and depth estimation [14][15]. This approach provides the model with more abstract and decision-friendly features compared with the RGB images. However, it may increase the unnecessary complexity of the driving model as the mediated perception is a high-dimensional world representation possibly containing redundant information compared with low dimensional driving commands [9].

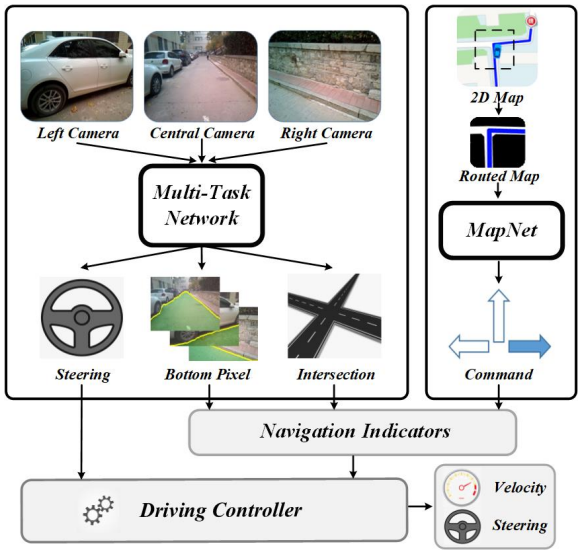

Fig. 1. Overview of the proposed method. Each image input is a separate channel and the outputs are fed into a designed controller to generate the final steering and velocity for robot navigation.

By contrast, behavior reflex approaches avoid handcrafted rules and use end-to-end models to map from raw RGB images to commands by learning from human driving demonstrations. Existing methods have showcased favorable performance in specific tasks [8][16][17][18]. However, such models work like a black box by completing a complex mapping within a single step, which may reduce the interpretability of the model and might not be trusted or confidently used by humans. Moreover, different human drivers may take different attentions when facing the same situation, which will confuse the model training.

In addition to the above two paradigms, direct perception has been studied recently. It directly predicts low-dimensional intermediate representations of the environment, such as distance to obstacles [19] and distance to the lane markings [20]. Instead of pursuing a complex parsing of the entire scenario, such representations are then fed into a designed driving controller to control the vehicle. It combines the advantages of the above two paradigms. However, as stated in [20], since the intermediate representations are diverse and difficult to obtain in practice, most studies are restricted to simulated environments only.

In this paper, we present a novel method which combines the advantages of the above paradigms in a multi-view manner. Most existing work only utilizes a monocular camera, which is usually inadequate to learn a safe driving model due

This work was funded by the National Natural Science Foundation of China under Grant 61991411 and Grant U1913204, the National Key Research and Development Plan of China under Grant 2017YFB1300205, the Shandong Major Scientific and Technological Innovation Project 2018CXGC1503, the Young Taishan Scholars Program of Shandong Province No. tsqn20190929, and the Qilu Young Scholars Program of Shandong University No. 31400082063101.

All authors are with the School of Control Science and Engineering, Shandong University, Jinan, China.
†These authors contributed equally to this work.
*Corresponding author: Wei Zhang (Email: davidzhang@sdu.edu.cn)

to the limited field of view [17][8][21]. By contrast, multiple cameras can see if other objects are coming from multiple sides and thus provide more information for making safer decisions. As shown in Fig. 1, the proposed multi-camera system includes one central camera and two side cameras. We also develop a multi-task network to learn the navigation skills from the three perspectives of observations. Besides the shared task of transforming input RGB images into mediated representations of the environment, in the first sub-task, we use the input of the central camera to directly regress an initial steering which performs well on lane keeping. In the second sub-task, we use the input of the side cameras to detect whether the vehicle has reached an intersection. The high-level commands are mapped from a 2D map with planned point-to-point route by the proposed MapNet. The result of the detection is then combined with the high-level commands to determine the travel direction.

Since the mediated representations of high dimension can hardly be used for direct decision making, we transform it into three low-dimensional indicators in a probabilistic form. The three indicators including the local, the global and the combined ones are generated through the combination of the high-level commands output by the MapNet, the detection of the obstacle by locating its bottom pixels and the results of intersection detection output by the multi-task network. Moreover, a designed driving controller is used to implement the command including steering and velocity based on the network outputs and the indicators. Extensive results on a robotic vehicle prove that our autonomous navigation approach outperforms existing methods in both obstacle avoidance and goal-directed navigation tasks.

II. RELATED WORK

Mediate perception approaches have been well developed via extensive research on various sub-tasks of autonomous navigation, such as image segmentation [22][23][24] and object detection [1][25][26]. These approaches are relatively interpretable and safe due to their modularity, and thus popular in industry. However, most of the current works focused on data-driven behavior reflex approaches without handcrafted rules.

Behavior reflex approach, first proposed in [27], directly maps driving actions from raw sensory input. Recently, Muller et al. [16] and Bojarski et al. [8] demonstrated the effectiveness of neural networks in predicting action, for specific tasks, such as lane keeping and obstacle avoidance. Loquercia et al. [17] introduced an approach to learning control commands from the data recording human drivers' behaviors for lane keeping and predicting the collision probability for collision avoidance. Similarly, Codevilla et al. [28] proposed a conditional imitation learning approach which can learn from human demonstration with the input of high-level command, which enables the vehicle to traverse intersections in desired direction. However, this single-step mapping approach lacks transparency [20].

Another line of work treated autonomous navigation as a visual task in which the vehicle is controlled by a designed controller based on a small number of indicators. Sauer et al. [20] and Chen et al. [9] demonstrated good results when applying this approach to structured environment driving in an open-source simulator. Kouris et al. [19] proposed to use the distance-to-collision in three directions for decision making and performed well in indoor environments. Unfortunately, this approach is unsuitable for outdoor environments.

Most existing methods including the aforementioned ones are based on a single forward-looking camera, while multi-camera systems play a vital role in providing a wider field of view for the decision making in navigation tasks. Multi-camera based systems have been studied recently. For example, Baek et al. [24] used multiple fisheye cameras for surrounding scene understanding. Deng et al. [29] introduced a restricted deformable convolution method for semantic segmentation using surround view cameras. Also, Amini et al. [30] and Hecker et al. [31] presented methods using a neural network to construct direct mapping from images of multiple cameras and a routed map to behavior. However, the increase of image input may lead to information redundancy, which increases the difficulty of model training. Meanwhile, as studied in [32], naively incorporating all camera features may lead to the problem of over-reliance. By contrast, our method treats each input from an individual camera as a separate channel. The output of each channel is then used to implement control commands, ensuring that the information from each view is well explored for decision making.

III. METHOD

This section elaborates each of the three parts of the proposed method. In the first part, we introduce the architectures of different networks. These networks are used for initial steering prediction, scene perception and high-level command mapping. In the second part, we introduce the method for extracting indicators excluding the predicted steering from the outputs of the networks. Finally, a controller is developed to receive the above indicators and the predicted steering to generate final control commands.

A. Network Architecture

We propose a multi-task network aiming at the detection of the free drivable area, and the steering and intersection inference via different cameras. The central camera is used for steering prediction, and side cameras for intersection detection. We also propose a network, namely MapNet, for classifying the high-level command from the routed map. As shown in Fig. 2, the multi-task network has a shared encoder for two tasks to extract features from the input image. After the feature extraction, a convolutional layer with ReLU activation, an upsampling layer and a softmax layer are used to detect the traversable area by locating the bottom pixel of the obstacle corresponding to the column containing that pixel, highlighted as the yellow lines in Fig. 2. Considering the efficiency of deploying an onboard platform with multiple cameras, unlike the network proposed in [24], we use the MobileNetV2 [33] branching at the fifth "bottleneck" as the encoder and replace the decoder part with

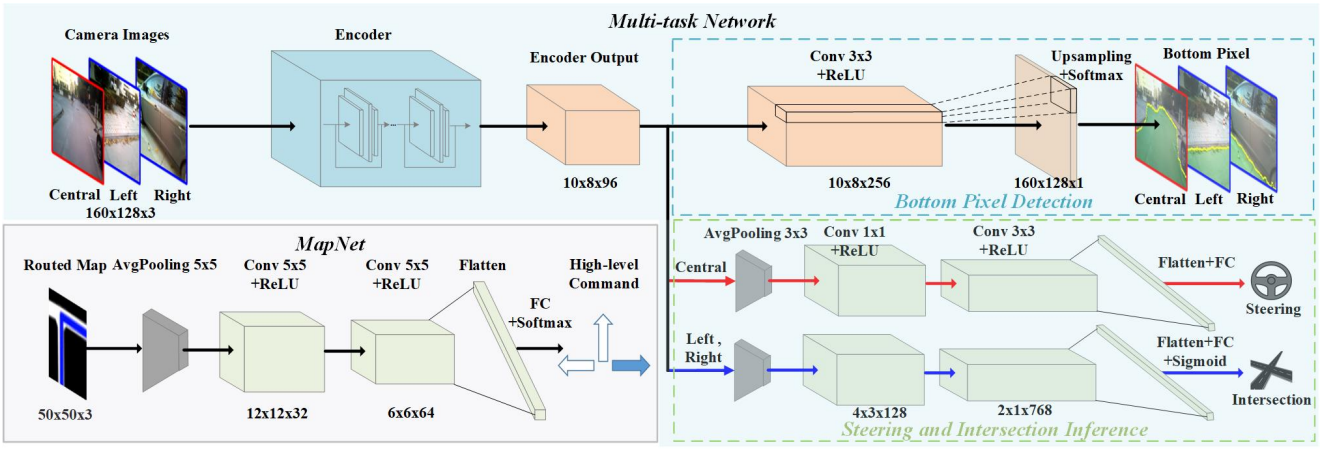

Fig. 2. Illustration of the proposed network architecture. The multi-task network receives images from different cameras separately and outputs the results of bottom-pixel detection, intersection detection and initial steering commands. The MapNet receives routed map images and outputs high-level command.

a single dense upsampling convolution. Besides, the network for predicting the initial steering and detecting the road intersection has one average pooling layer, two convlutional layers with ReLU activation and a fully connected layer. For intersection detection, a sigmoid layer is employed at the end of the network. The MapNet is a simple network structurally similar to the multi-task network for the steering and intersection inference while a softmax layer is used to classify the high-level command.

We use the binary cross-entropy loss (BCE) and the mean-squared loss (MSE) to train the networks detecting intersection and predicting steering. Softmax cross-entropy loss (SCE) is used for bottom-pixel detection and high-level command classification. The loss functions are shown below:

$$\mathcal{L}_{BCE}(t,y) = -\frac{1}{N}\sum_{i=1}^{N}[t_i log(y_i) + (1-t_i) log(1-y_i)], \quad (1)$$

$$\mathcal{L}_{MSE}(t,y) = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2, \quad (2)$$

$$\mathcal{L}_{SCE}(t,y) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} t_i(k) log(y_i(k)), \quad (3)$$

where t is the labels of the training dataset, y is the output of network and N is the number of samples. K is the number of class labels in the multi-class classification task. For bottom-pixel detection task, K is the height of the input image, while for high-level command classification K is 3. During the training of multi-task networks, simple joint optimization will lead to a convergence problem due to the different loss gradients generated by different tasks [17]. Moreover, finding appropriate weights between different losses is a complex and difficult task. Hence, we propose to train the tasks separately. We first train the bottom-pixel detection task and then freeze the parameters to serve as an initializing feature extractor to train other tasks.

In the implementation, the proposed multi-task networks and the MapNet are trained using images with the resolutions of 160×128 and 50×50 respectively and the batch size of 32. For data augmentation, we use random brightness, and add salt-and-pepper noise on all camera data. Besides, we use horizontal flip on the bottom-pixel detection data. We use the Adam optimizer [34] to train our models with an initial learning rate of 10^{-4}.

B. Navigation Indicators

The robotic vehicle should be able to keep moving along the road while following the route command and avoid obstacles such as pedestrians and cars. To satisfy the requirements for real-world driving, we set the following rules to generate the navigation indicators: i) The vehicle should move in the middle the road if there are no obstacles ahead; ii) When reaching an intersection, the vehicle should be able to turn according to high-level command; iii) The vehicle should be able to select the best direction to bypass obstacles under the observation of the surrounding environment.

In accordance with the above rules, we mainly use predicted steering for road keeping. Moreover, we propose two types of indicators of global and local navigation. Finally, these two types of indicators are integrated into combined indicators then used for the last two rules.

1) Local Navigation Indicators: We propose to extract the local indicators based on bottom-pixel information to evaluate the environment around the vehicle. We use the position of bottom-pixel as approximate distance measure to obstacles. We first normalize it to [0,1]. In the normalization, we do not consider the upper 1/4 image as it is usually covered by remote buildings or sky. We then divide the images into a series of overlapped windows. As shown in the left image of Fig. 3, the red and blue windows are used to monitor the state of the vehicle's front and left side respectively. The left half of the central image and the left image are used together to describe the overall traversable area on the left side. W_c, W_{ls}^1 and W_{ls}^2 are the normalized bottom-pixel sets in the central, left and right windows respectively. Moreover, as shown in the right image of Fig. 3, a sliding window is used to find the optimal area for lateral vehicle movement based on the maximum mean value of the

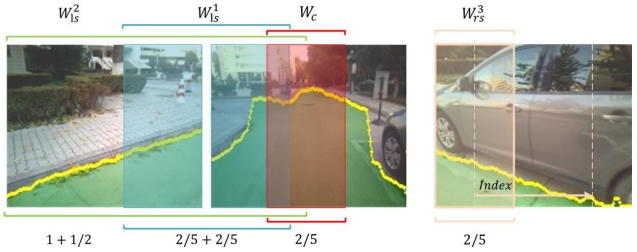

Fig. 3. Left: Overlapped windows for extracting indicators. Right: A sliding window for finding optimal area for lateral vehicle movement. The number at the bottom gives the ratio of the window width to the image width.

normalized bottom-pixel set in the corresponding window. W_{rs}^3 is the normalized bottom-pixel set corresponding to the output window. Similarly, we also have W_{rs}^1, W_{rs}^2 on the right side and W_{ls}^3 on the left side. i_{ls} and i_{rs} denote the indexes of the corresponding sliding windows.

According to the windows and normalized bottom-pixel, we propose one longitudinal safety indicator (L_c) based on the central camera to evaluate the safety state of local longitudinal movement. Also, two lateral safety indicators (L_{ls} and L_{rs}) are proposed based on the central camera and the side cameras, which are used to evaluate the safety state of local lateral movement. Considering that the obstacle closest to the vehicle will bring the highest risk, these indicators are defined as follows:

$$L_c = min(1, \frac{1}{1+e^{-\alpha\times(min(W_c)-\beta)}}), \tag{4}$$

$$L_{ls} = min(1, \frac{1}{1+e^{-\alpha\times(min(W_{ls}^1)-\gamma)}} + (1-v)), \tag{5}$$

$$L_{rs} = min(1, \frac{1}{1+e^{-\alpha\times(min(W_{rs}^1)-\gamma)}} + (1-v)), \tag{6}$$

In the experiments, we set $\alpha = 20$, $\beta = 0.25$, and $\gamma = 0.15$ empirically. v is the velocity of the robotic vehicle. The longitudinal safety indicator is only related to the distance to the nearest obstacle, while the lateral safety indicators are also related to the vehicle's velocity v as we allow the vehicle to make careful turns at a low speed with obstacles around.

2) Global Navigation Indicators: We use $P(\text{IS})$ to denote the intersection detection result, a probability which represents how possible the corresponding lateral direction is an intersection. Then, the high-level command $cmd \in [turn\ left,\ go\ forward,\ turn\ right]$ mapped from the routed map is used to select the driving direction at the intersection. The global indicators for the left and right sides are set as follows:

$$\begin{cases} G_{ls} = P(\text{IS}),\ G_{rs} = 0 & cmd = turn\ left \\ G_{ls} = 0,\ G_{rs} = P(\text{IS}) & cmd = turn\ right \\ G_{ls} = 0,\ G_{rs} = 0 & cmd = go\ forward \end{cases}. \tag{7}$$

3) Combined Indicators: We integrate the two types of indicators and propose the three combined indicators below:

$$M_c = \overline{W_c} \times L_c, \tag{8}$$

$$M_{ls} = \begin{cases} \overline{W_{ls}^3} \times (1-L_c) \times L_{ls} & L_c < 0.5 \\ \overline{W_{ls}^3} \times G_{ls} \times L_{ls} & otherwise \end{cases}, \tag{9}$$

Algorithm 1 Driving Controller for Navigation

Input: $L_c, L_{ls}, L_{rs}, M_c, M_{ls}, M_{rs}, \overline{W_{ls}^2}, \overline{W_{rs}^2}, i_{ls}, i_{rs}, s_p$
Output: $velocity, steering$

repeat
  $\vec{M_c} \leftarrow M_c,\ \vec{M_{ls}} \leftarrow \{M_{ls}, n_{ls}\},\ \vec{M_{rs}} \leftarrow \{M_{rs}, n_{rs}\}$
  $velocity = L_c \times V_{max}$:
  if M_{ls} or M_{rs} **is not** 0 **then**
    if $L_c < \sigma$ **then**
      Choose the direction $i \in \{ls, rs\}$ for obstacle avoidance according to $max\{\overline{W_{ls}^2}, \overline{W_{rs}^2}\}$
    else
      Choose the direction $i \in \{ls, rs\}$ for navigation according to $max\{M_{ls}, M_{rs}\}$
    The resultant vector's angle $\theta_t = \theta(\vec{M_c} + \vec{M_i})$
    Map the angle to the steering $s_t = \theta_t / 90°$
    $steering = s_t$
  else
    Choose the direction $i \in \{ls, rs\}$ based on s_p
    Ensure safe turn $steering = s_p \times L_i$
until reach destination

$$M_{rs} = \begin{cases} \overline{W_{rs}^3} \times (1-L_c) \times L_{rs} & L_c < 0.5 \\ \overline{W_{rs}^3} \times G_{rs} \times L_{rs} & otherwise \end{cases}, \tag{10}$$

where $\overline{W_c}$, $\overline{W_{ls}^3}$ and $\overline{W_{rs}^3}$ are the mean values of W_c, W_{ls}^3 and W_{rs}^3 respectively. So the above combined indicators take into account both the local safety and the global moving direction of the robot. Since these indicators are low-dimensional, it is easy to design a controller based on them.

C. Control Policy

The proposed indicators with the predicted steering s_p are fed into a designed driving controller to generate control policies including velocity $v \in [0, 1]$ m/s and the final steering $s \in [-1, 1]$ rad/s. According to these indicators, the controller yields different control policy for different situations. This allows safe navigation and more interpretable decision making by opting for moving towards the desired direction or that with the largest amount of traversable space.

The whole controlling policy is described in Algorithm 1. The vehicle velocity v is defined to be proportional to the local indicators of the longitudinal movement L_c. V_{max} denotes the maximum speed of the vehicle. At each time step, the controller judges if the side cameras are needed to assist the decision making based on the combined indicators of different cameras. With the aid of the side cameras, we use a motion planning method to generate steering command as an alternative to the initial steering in specific situations, like reaching an intersection or facing obstacles. We first construct a set of vectors $\{\vec{M_{ls}}, \vec{M_c}, \vec{M_{rs}}\}$ based on the the combined indicators. Instead of using fixed windows and vector angles like [19], we used the indexes i_{ls}, i_{rs} of the optimal sliding window to adjust the angle of the vectors for motion smoothing. These vectors are towards the left, central and right directions, corresponding to $[-90°, -30°]$, $0°$, and $[30°, 90°]$ respectively. If the longitudinal safety indicator

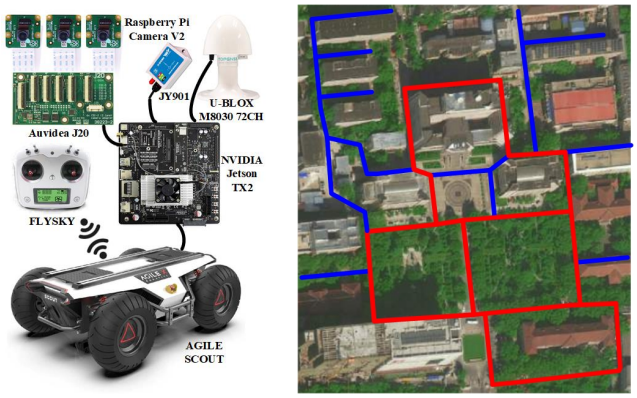

Fig. 4. Left: Our mobile platform. Right: Experimental environment where the red and the blue routes are for data collection and testing respectively.

L_c is less than the threshold σ set to 0.5, the direction for obstacle avoidance is selected as the one with the most traversable area based on $\overline{W_{ls}^2}$ and $\overline{W_{rs}^2}$, the means of W_{ls}^2 and W_{rs}^2 respectively. Otherwise, the direction is determined by the maximum combined indicators. As an alternative to the predicted steering, s_t is determined by the angle of the resultant vector derived from $\vec{M}_c$ and $\vec{M}_i$. In other cases, the predicted steering s_p is used to control the vehicle moving along the road. Meanwhile, the lateral safety indicators L_{ls}, L_{rs} are used to adjust the steering to prevent lateral collision.

IV. EXPERIMENTAL RESULTS

In this section, we conduct experiments of robot navigation in the real-world environments to demonstrate the superiority of the proposed model over the baseline methods.

A. System Setup

Physical Setup: The physical system is illustrated in the left image of Fig. 4. All of the components except for the remote control are mounted to the vehicle, AGILE SCOUT. We run the Robot Operating System (ROS) and Ubuntu on the NVIDIA Jetson TX2. We equip the truck with three raspberry Pi RGB cameras driven by the J20 module. The field of view of all three cameras is 60 degrees, and the adjacent angle is 60 degrees too. Besides, we use a third-party map API to obtain 2D maps with navigation routes based on GPS information. In order to remove the redundant information, we crop and binarize the map.

Testing Scenario: As shown in the right image of Fig. 4, we test our method with the vehicle running in a campus environment. We collect data from human demonstrations for a total of 45K images from all three cameras as well as the corresponding map images supplying the route information. These images are used for bottom pixel detection, high-level command classification, and steering prediction which only uses the images from the central camera. Due to the lack of positive samples of intersection detection, we also expand the positive samples by including some images from the central camera, leading to a set of total 12K images used for intersection detection. In each task, 20% data are used for validation.

TABLE I

COMPARISON OF VARIOUS BACKBONES USING DIFFERENT METRICS

Backbone	Resnet8	Resnet50	Inception-ResNet-V2	MobileNetV2
RMSE	0.168	0.099	0.102	0.108
Avg. Accuracy	91.72%	96.68%	96.17%	94.74%
MAE	4.83	2.33	2.65	2.88
FPS	13.52	4.97	3.88	8.92

TABLE II

QUANTITATIVE COMPARISONS BETWEEN OUR METHOD AND BASELINES. NO. UI AND NO. MI DENOTE THE NUMBER OF USER INTERVENTIONS AND MISSED INTERSECTIONS RESPECTIVELY.

Environment	Model	No Map Task		With Map Task		
		Driving Length	Driving Time	No. UI	No. MI	Driving Time
Simple	DroNet	$100m$	$104s$	2.5	1.7	$144s$
	PilotNet	$117m$	$115s$	2.3	1.3	$138s$
	MVE2E	$123m$	$120s$	0.6	1	$121s$
	SV	$124m$	$120s$	0.5	1.1	$126s$
	Ours	**136m**	**120s**	**0**	**0**	**110s**
Complex	DroNet	$46m$	$79s$	6.5	1.9	$177s$
	PilotNet	$53m$	$84s$	5.3	1.5	$146s$
	MVE2E	$67m$	$88s$	5.1	1.3	$132s$
	SV	$87m$	$104s$	2.2	1.4	$135s$
	Ours	**128m**	**120s**	**0**	**0.4**	**122s**

B. Backbone Comparison

In this section, we use different networks [33][35][36] as the backbone where they are modified to meet the dimensional requirements for up-sampling. To compare the model variants using different encoder networks in terms of both accuracy and speed, we report root-mean-squared error (RMSE), average accuracy, mean absolute error (MAE) and frame per second (FPS) in Table I.

We use RMSE to perform steering prediction. For intersection detection and high-level command classification, average classification accuracy is used to evaluate the performance. For bottom-pixel detection task, we adopt the MAE metric, which represents the mean pixel displacement between the ground truth and the prediction. The number of FPS onboard is used to reflect the computational efficiency of the model.

As shown in Table I, due to the deeper network structure and the residual modules, ResNet-50 and Inception-ResNet-V2 achieved better performance, but the computational inefficiencies made them difficult to be deployed on mobile platforms. A high computational efficiency is vital for robot navigation as it ensures that the vehicle can respond to emergency in time. Although ResNet-8 is a residual network with higher computational efficiency, its shallow network architecture is unsuitable for the complicated pixel-wise detection task. Considering the balance between computational efficiency and performance, we finally use MobileNetV2 as the encoder.

C. Baselines and Tasks

In the experiments, we compare our method with three state-of-the-art methods: DroNet [17], PilotNet [8] and MVE2E [30]. Both DroNet and PilotNet are a single-view models. It is worth mentioning that PilotNet uses three cameras for training while only the central camera is used for navigation. Similar to ours, MVE2E is a multi-view

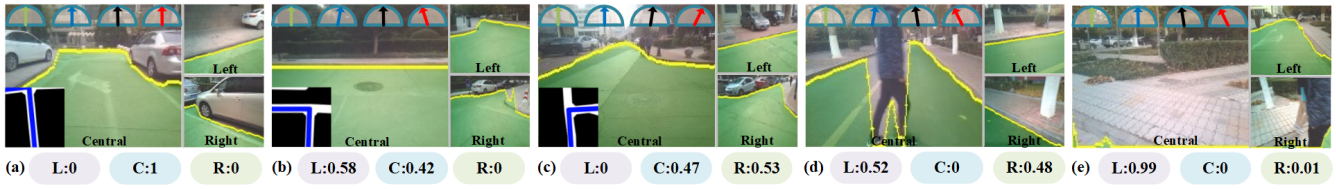

Fig. 5. Samples in different scenarios which demonstrate comparisons of control commands between our system and other methods (Green: DroNet, Blue: PilotNet, Black: MVE2E, Red: Ours). The numbers under the images are the decision-making ratios of different cameras.

model which is an updated version of PilotNet. However, differing from our method, it merges the images from the three cameras in a naive way, i.e., directly fusing the image features from the three cameras, which leads to over-reliance on the central images. Besides, we have introduced an ablated version of our method, SV. The only difference is that SV uses only the central camera for training and testing. To train the obstacle avoidance models of PilotNet and MVE2E, we additionally collect some images with obstacles and corresponding steering for bypassing obstacles. Meanwhile, these additional data are also used to train the collision inference model of DroNet.

For comparison, we define the following two tasks to evaluate the performance of the baselines and our method.

No Map Task: There is no planned route and the termination condition is that a collision occurs or a preset maximum duration (120s) is reached.

With Map Task: A routed map is added as extra navigation information, which requires the robot to reach the designated destination according to the planned route.

The above two tasks are both conducted in simple and complex scenarios. The simple scenario only contains some static obstacles in the environment, while the complex one includes dynamic obstacles like pedestrians or artificially added barricades. The two tasks were both tested 10 times and the average performance were reported. We evaluate the performance for No Map Task in terms of driving length and time. For With Map Task, we have added two additional metrics: User Interventions and Missed Intersections. The number of user interventions refers to the times that an error occurred during the testing, e.g. hitting an obstacle which requires a manual reset. Missed Intersection denotes the number of intersection missed.

D. Quantitative and Qualitative Evaluations

As shown in Table II, in No Map Task, when there is no obstacle in the middle of the road, both our method and baselines can drive a long distance, indicating that all approaches can well complete the lane keeping tasks in a simple environment. However when encountering dynamic obstacles, DroNet, PilotNet and SV are limited by a narrow field of view and often hit the obstacles. Their driving length and time are less than ours. MVE2E also fails to avoid dynamic obstacles due to the naive merging of the images of three cameras and thus cannot drive a long distance.

In With Map Task, map information was added as a high-level planner to complete the point-to-point navigation task. However, as shown in Table II, due to the deviation of GPS positioning, DroNet, PilotNet and SV often miss some intersections for turning. These methods rely on a high precision of GPS positioning as the visual information supplied by a single camera is not sufficient to find the intersections. MVE2E benefits from multiple cameras and thus misses less intersections. Our method does not only use multiple cameras to take images, but also includes a specific network handling the images captured by the side cameras for intersection detection. Consequently, our method requires the least interventions and misses the least intersections.

Fig. 5(a) shows that all of the methods worked well when the robot just needs to go forward. However, as shown in Fig. 5(b) and (c), only our method enabled the robot to turn accurately at the intersections while the competing methods apparently missed the intersections. In Fig. 5(d), there is a dynamic obstacle in front of the robot while passable areas are available on both sides. DroNet often suffered from a standstill when facing the obstacle. PilotNet ended with constantly swaying from side to side due to the narrow field of view. The steering angle predicted by MVE2E is too small to avoid obstacles as it cannot decide which side to pass through. In Fig. 5(e), the initial position has obstacles blocking the front and the right side of the robot, but there is a passable area on the left side. Similarly, both DroNet and PilotNet hit the obstacle as they cannot detect the passable area due to the lack of the information from both sides. MVE2E cannot accurately estimate the turning angle and cause a collision while it can find the passable area due to its multi-camera setup. In contrast, our method can accurately identify which side is accessible and successfully bypass the obstacle. It is worthing noting that instead of using a constant velocity for obstacle avoidance, our method can yield a proper velocity based on the saftey state of the vehicle as shown in Algorithm 1.

Besides, to evaluate the contribution of the information from each camera to the final decision, we define the decision-making ratio, which is a ratio of the global navigation indicators of each camera to the sum of them. As shown in Fig. 5, in different scenarios, the contributions of all three cameras have different weights for the decision making, indicating that each camera takes effect. For instance, in Fig. 5(a), since there is no obstacle on the road, the contribution of the two side cameras is 0 and thus the vehicle drives only relying on the information provided by the central camera. In comparison, in Fig. 5(d), due to the pedestrian who blocks the road, our method

make the decision of turning left based on the information collected by the side cameras where the left side camera has a slightly higher contribution as it has more traversable area than the right side (L:0.52 *vs* R:0.48). Please refer to `https://vsislab.github.io/arnmcp/` for the video demonstration of robot navigation in the real world.

V. CONCLUSIONS

In this paper, we proposed a novel deep learning framework based on multi-camera perception to enable the smooth and safe autonomous navigation for a robot in the real world. Our navigation indicators greatly enhanced the robot's ability of path planning and obstacle avoidance. Importantly, we adopted multi-task learning approach that combines behavior reflex, mediated perception and direct perception to generate optimal control policies. Extensive navigation experiments were conducted in various real-world scenes to demonstrate the effectiveness of the proposed method.

REFERENCES

[1] P. Li, X. Chen, and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7644–7652.

[2] W. Zhang, Q. Chen, W. Zhang, and X. He, "Long-range terrain perception using convolutional neural networks," *Neurocomputing*, vol. 275, pp. 781–787, 2018.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] D. Fox, S. Thrun, and W. Burgard, "Probabilistic robotics," 2005.

[5] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Simultaneous perception and path generation using fully convolutional neural networks," *arXiv preprint arXiv:1703.08987*, 2017.

[6] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov, "Combining neural networks and tree search for task and motion planning in challenging environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 6059–6066.

[7] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2174–2182.

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[9] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[10] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017.

[11] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-scnn: Gated shape cnns for semantic segmentation," in *IEEE International Conference on Computer Vision*, 2019, pp. 5229–5238.

[12] J. Mao, T. Xiao, Y. Jiang, and Z. Cao, "What can help pedestrian detection?" in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3127–3136.

[13] A. D. Pon, J. Ku, C. Li, and S. L. Waslander, "Object-centric stereo matching for 3d object detection," *arXiv preprint arXiv:1909.07566*, 2019.

[14] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *IEEE International Conference on Computer Vision*, 2019, pp. 3828–3838.

[15] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2800–2810.

[16] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. MIT Press, 2006, pp. 739–746.

[17] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[18] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 8248–8254.

[19] A. Kouris and C.-S. Bouganis, "Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1–9.

[20] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," *arXiv preprint arXiv:1806.06498*, 2018.

[21] S. Chowdhuri, T. Pankaj, and K. Zipser, "Multinet: Multi-modal multi-task learning for autonomous driving," in *IEEE Winter Conference on Applications of Computer Vision*, 2019, pp. 1496–1504.

[22] S. Tsutsui, T. Kerola, S. Saito, and D. J. Crandall, "Minimizing supervision for free-space segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 988–997.

[23] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3146–3154.

[24] J. Yeol Baek, I. Veronica Chelu, L. Iordache, V. Paunescu, H. Ryu, A. Ghiuta, A. Petreanu, Y. Soh, A. Leica, and B. Jeon, "Scene understanding networks for autonomous driving based on around view monitoring system," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 961–968.

[25] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1–8.

[26] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2147–2156.

[27] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[28] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 1–9.

[29] L. Deng, M. Yang, H. Li, T. Li, B. Hu, and C. Wang, "Restricted deformable convolution-based road scene semantic segmentation using surround view cameras," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[30] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 8958–8964.

[31] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in *European Conference on Computer Vision*, 2018, pp. 435–453.

[32] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah *et al.*, "Urban driving with conditional imitation learning," *arXiv preprint arXiv:1912.00177*, 2019.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.