Deep Reinforcement Learning for Robot Collision Avoidance With Self-State-Attention and Sensor Fusion

Yiheng Han, Irvin Haozhe Zhan, Wang Zhao, Jia Pan, *Senior Member, IEEE*, Ziyang Zhang, Yaoyuan Wang, and Yong-Jin Liu, *Senior Member, IEEE*

***Abstract*—3D LiDAR sensors can provide 3D point clouds of the environment, and are widely used in automobile navigation; while 2D LiDAR sensors can only provide point cloud in a 2D sweeping plane, and then are only used for navigating robots of small height, e.g., floor mopping robots. In this letter, we propose a simple yet effective deep reinforcement learning (DRL) method with our self-state-attention unit and give a solution that can use low-cost devices (i.e., a 2D LiDAR sensor and a monocular camera) to navigate a tall mobile robot of one meter height. The overrall pipeline is that we (1) infer the dense depth information of RGB images with the aid of the 2D LiDAR sensor data (i.e., point clouds in a plane with fixed height), (2) further filter the dense depth map into a 2D minimal depth data and fuse with 2D LiDAR data, and (3) make use of DRL module with our self-state-attention unit to a partially observable sequential decision making problem that can deal with partially accurate data. We present a novel DRL training scheme for robot navigation, proposing a concise and effective self-state-attention unit and proving that applying this unit can replace multi-stage training, achieve better results and generalization capability. Experiments on both simulated data and a real robot show that our method can perform efficient collision avoidance only using low-cost 2D LiDAR sensor and monocular camera.**

***Index Terms*—Deep reinforcement learning, robot Collision avoidance, self state attention, sensor fusion.**

I. Introduction

ROBOT navigation is an important topic in robotic research. Both static and dynamic environments have been considered for robot navigation. Some early methods only handle static objects [1], [2] or simply treat moving objects as static objects at a specific point in time [3], [4]. Other representative navigation methods only take actions for the next step (in a small time interval) of the robot without considering the global trajectory [5]–[7], which often results in unsafe and unnatural moving trajectories.

Multi-agent robot navigation such as the optimal reciprocal collision avoidance (ORCA) [6], and its variants including ORCA-DD [8] and NH-ORCA [9] have been proposed for multi-agent systems working in dynamic scenarios. However, these methods suppose each robot to have perfect sensing about the surrounding agents' positions, velocities, and shapes, which lead to a poor generalization in complex scenarios.

Recently, deep learning techniques [10]–[14] have been introduced into robot navigation, showing its effectiveness in this problem. To better utilize deep learning methods, robot navigation is abstracted into a decision-making problem in the action space with collision avoidance as constraints, which can be effectively handled by deep reinforcement learning (DRL) strategy [15]–[18]. DRL can generate a large amount of simulation data to train the model without labeled data, which makes the trained model perform well in dynamic scenarios with good global trajectories.

The DRL methods often use RGB/RGBD image data or point cloud data from LiDAR sensors as input. For image data input, it is usually difficult for DRL to transfer the trained models to real-world environments because of the difference between simulation data and real-world data [16], [17], [19]. For point cloud data, due to the high cost of 3D LiDAR sensors, most robots of small height (e.g., floor mopping robots) have used 2D LiDAR sensors, which can only output sparse point cloud lying in a 2D sweeping plane. However, this kind of point cloud is not suitable for a high robot such as the one shown in Fig. 1.

In this letter, we propose a concise and effective DRL method with our self-state-attention unit and give a solution that only uses low-cost 2D LiDAR sensor and monocular camera for navigation of a tall robot. The overall pipeline is to infer the dense depth information of RGB images by making use of 2D LiDAR sensor data, and then conversely use the estimated dense depth to augment the 2D LiDAR data, by compressing the estimated dense depth map into 2D minimal depth data and fusing with 2D LiDAR data. Finally, inspired by the problem that the weight of

Manuscript received February 24, 2022; accepted May 6, 2022. Date of current version June 6, 2022. This letter was recommended for publication by Associate Editor C. Paxton and Editor H. Kurniawati upon evaluation of the reviewers' comments. This work was supported in part by the Natural Science Foundation of China under Grant 61725204, in part by Tsinghua University Initiative Scientific Research Program under Grant 20211080093, and in part by the Science and Technology Department of Jiangsu Province, China. *(Yiheng Han and Irvin Haozhe Zhan are co-first authors.) (Corresponding author: Yong-Jin Liu.)*

Yiheng Han, Irvin Haozhe Zhan, Wang Zhao, and Yong-Jin Liu are with the BNRist, MOE-Key Laboratory of Pervasive Computing, Department of Computer Science and Technology, Tsinghua University, Beijing 100190, China (e-mail: hyh18@mails.tsinghua.edu.cn; zhanhz20@mails.tsinghua.edu.cn; thu zhaowang@163.com; liuyongjin@tsinghua.edu.cn).

Jia Pan is with the Department of Computer Science, University of Hong Kong,, Hong Kong (e-mail: panjia1983@gmail.com).

Ziyang Zhang and Yaoyuan Wang are with the Advanced Computing and Storage Laboratory, Huawei Technologies Company Ltd, Shenzhen 518129, China (e-mail: zhangziyang11@huawei.com; wangyaoyuan1@huawei.com).

Digital Object Identifier 10.1109/LRA.2022.3178791

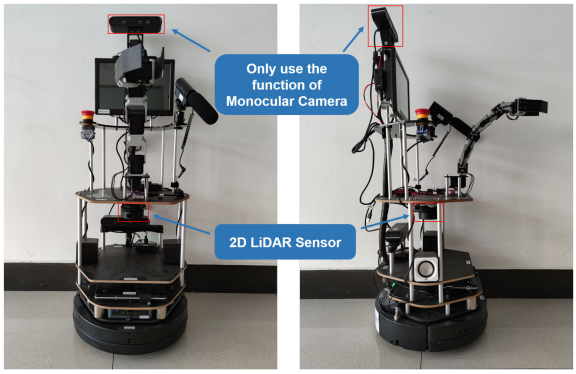

Fig. 1. A multi-functional robot of 1 m height used in this letter to test the navigation algorithm. This robot is equipped with a camera, a 2D LiDAR sensor, a robotic arm, a microphone that receives sound, an LCD screen, and a location sensor. We use the RGB images from the camera and the 2D LiDAR sensor to navigate the robot using the technique proposed in this letter.

reward is difficult to adjust in scenes of different complexity, we present a concise and effective self-state-attention DRL training scheme for collision avoidance, which can avoid complex multi-stage weight adjustment and has better generalization capability. Both simulation evaluation and real-world experiments are performed, showing the effectiveness of the proposed method.

II. Related Work

A. Robot Navigation With DRL

Robot navigation is mainly based on RGB/RGBD image data or point cloud information. Ma *et al.* [19] present a motion planner that takes RGB images as input and odometry as target reference. This method takes a long time to train and does not perform well enough in the real world. To overcome the gap between simulation and real-world scenarios, Bharadhwaj *et al.* [20] proposed a framework to combine simulation and real RGB image data. Generally, the difference between simulated and real image data (e.g., the significant difference in complex material property of object surfaces, rendering and shading of multiple point and area light sources, etc.) often results in a poor DRL performance on real-world robot navigation.

On the other hand, the geometry information in point cloud has small difference between simulated and real-world scenarios. Therefore, LiDAR sensor-based DRL has received considerable attention recently. Tai *et al.* [21] proposed a mapless motion planner by using a sparse 10D directional information as references, which was trained end-to-end through an asynchronous DRL method. Long *et al.* [16], [17] proposed a multi-scenario and multi-stage training framework which is based on DRL and PPO reinforcement learning method [22] trained in a set of complex environments for a large-scale robot system.

These point-cloud-based DRL methods are mainly applied to mobile robots with a low height, since the point cloud outputs by 2D LiDAR sensors are lying a 2D sweeping plane. For higher robots, 3D LiDAR sensors are required, which are expensive and in practice mostly used for autonomous driving cars. To tackle this problem, we propose a method using low-cost 2D LiDAR sensor and monocular camera, which jointly enable effective perception for downstream robot collision.

Recently, a lot of work start to use attention to model the interactions between robot and human or crowds [23]–[27]. These works tend to design interaction models to predict human actions and routes to place collisions with humans or crowds. The way they use attention to model interactions inspires us, but we think this may lead to a large number of invalid computations. Imagining a person navigating through a complex scene, attention is of course an important part of this task. However, we think it is more reasonable to combine the attention mechanism with the perception of the robot itself.

In this letter, inspired by the problem that the weight of reward is difficult to adjust in scenarios of different complexity and human-robot interaction attention, we design a novel self-state-attention unit for DRL module, which has better generalization capability while avoiding complex multi-stage weight adjustment.

B. Depth Completion

Our method uses the RGB image to complete the sparse point cloud from a 2D LiDAR sensor into a dense depth map, which is also known as the depth completion task. When the sparse point cloud information is absent, this task falls back to the depth estimation. Recently, deep learning methods have been widely used in the depth estimation problem. Eigen *et al.* [28] firstly proposed to estimate depth value from a single RGB image using deep convolution neural networks with two-scale architecture. Laina *et al.* [29] developed a deep residual network based on the ResNet. They proposed a novel up-convolution method to replace the traditional unpooling method, which achieved good results on both NYU Depth v2 [30] and Make3D datasets [31].

Since depth estimation from single RGB image alone is scale-ambiguous and unreliable, some researchers introduced sparse depth samples, which could be acquired from laser sensors or visual SLAM pipelines, and fused them with RGB images to achieve better results. Ma and Karaman [32] developed a deep regression model that takes both a sparse set of depth samples and RGB images as input and predicts a full-resolution depth image. Gansbeke *et al.* [33] improved the fusion process of the model. They used global and local information to estimate depth and learned confidence maps to guide the fusion of two results. Tang *et al.* [34] proposed GuideNet to exploit both RGB images and sparse 3D LiDAR data to get dense depth maps and achieved state-of-the-art results on KITTI in 2020 [35].

Some researchers try to exploit both 2D LiDAR and monocular image information to estimate depth. Liao *et al.* [36] suggested a novel method of input depth information called reference depth map, which is generated by extending 2D LiDAR in the image plane along the gravity direction. Lim *et al.* [37] proposed a multi-stage neural network to predict a dense depth map. To verify the effectiveness of the model under real conditions, they acquired actual 2D LiDAR data and RGB images and demonstrated good results.

In this letter, motivated by the rapid progress of deep depth estimation, we design a novel fusion algorithm which could

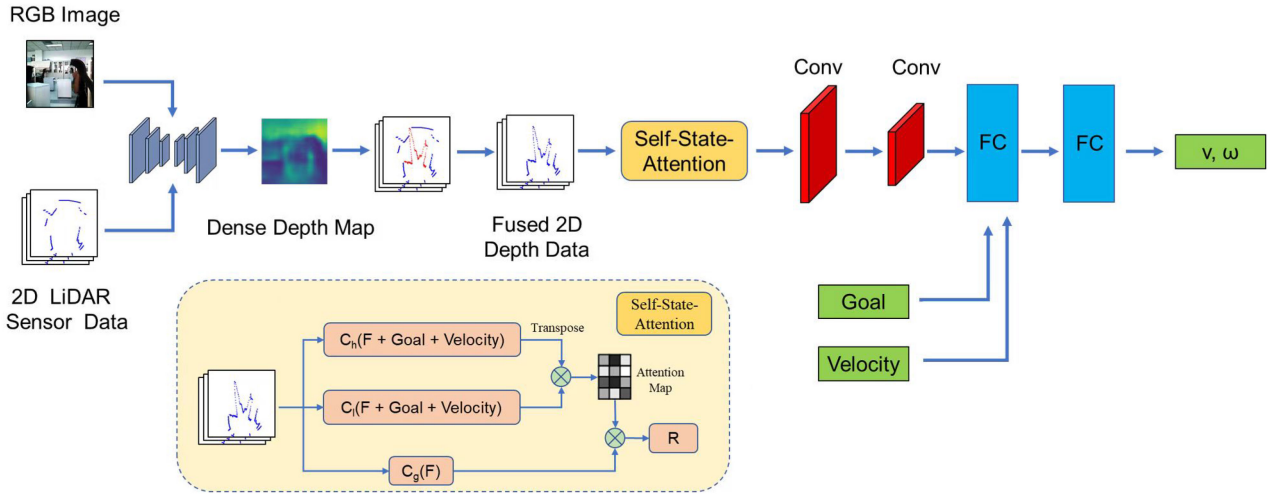

Fig. 2. Our framework of robot collision avoidance by fusing RGB images and 2D LiDAR sensor data. We first infer the dense depth information of RGB images using 2D LiDAR sensor data, and then use the predicted depth map to generate a filtered 2D minimal depth data. The 2D minimal depth data is combined with 2D LiDAR data to get the fused 2D depth data. After that, our novel self-state-attention unit is applied to remap the feature according to the importance. Finally, we input the fused data to our trained model and get the action decision, where v and ω are linear and angular velocities.

significantly augment the perception information used for robot navigation. Specifically, we first use both 2D LiDAR sensor data and RGB image data to estimate dense depth map, and then compress the predicted depth map into a 2D minimal depth data, fuse with 2D LiDAR data, to get much improved depth sensing information for a DRL-based robot navigation.

III. Method

We first recover the dense depth information of an RGB image $I^{H\times W}$ with the aid of 2D LiDAR sensor data (Section III-A), where H and W are the height and width of the image. Then we compress the recovered depth image into a 2D minimal depth data $L^{1\times W}$ after filtering out the ground information, where each number $L(i)$ is the minimal depth value of the i-th image column. The 2D minimal depth data is then fused with 2D LiDAR data (Section III-B).

Finally, we propose a simple yet effective DRL framework for robot collision avoidance with our novel self-state-attention unit (Section III-C) based on partially observable sequential decision making problem [16]. The whole pipeline of our method is shown in Fig. 2.

A. Depth Estimation

We infer the dense depth map of an RGB image $I^{H\times W}$, with the aid of 2D LiDAR sensor data P. Each point $p_l = (x_l, y_l, z_l)^T \in P$ has the same height. First we transfer the data from the LiDAR coordinate system to camera coordinate system by

$$p_c = (x_c, y_c, z_c)^T = Rp_l + T, \tag{1}$$

where R is the rotation matrix and T is the translation vector. Next, we project the points into the image plane, leading to the pixel coordinate $(u, v)^T$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}, \tag{2}$$

where f_x and f_y are camera focal lengths, and $(c_x, c_y)^T$ is the principal point. Note that for those points lying in the field of view (FOV) of 2D LiDAR sensor that are outside the camera FOV, they are filtered out. Finally, given an RGB image and a set of corresponding sparse 3D reconstructed points, we adapt a variant of the sparse-to-dense method [32] to obtain a dense depth map.

In our training processing in Section IV-A, we use simulated data. For this kind of data, we transfer the pixel data (u, v) (with known depth z_c) into camera coordinate system (x_c, y_c, z_c), using the formula

$$x_c = \frac{u - c_x}{f_x} z_c, \quad y_c = \frac{v - c_y}{f_y} z_c. \tag{3}$$

Then we select a set of 3D points of fixed height value to simulate the 2D LiDAR sensor data.

B. 2D Minimal Depth Data Generation and Fusion

The original 2D LiDAR sensor data only has sparse depth information at a fixed height, which is obviously not enough for the collision avoidance task of robots with a relatively tall height. If the 2D LiDAR has a height close to (or far away from) the ground, the robot will easily collide with tall (or short) obstacles. To address this problem, we convert the reconstructed depth map $M^{H\times W}$ into a 2D minimal depth data $\widetilde{L}^{1\times W}$, where H and W are the height and width of the map. Assume that the image plane of the camera is perpendicular to the ground and the camera is on top of the robot, which means the collision can only happen in $(0, \frac{H}{2})$. We set each number $\widetilde{L}(i)$ to be the

minimal depth value in the i-th column $M(j,i)$ of the depth map, $i = 1, 2, \ldots, W$. $j = 1, 2, \ldots, \frac{H}{2}$. Note that the depth values on the ground are misleading for 2D minimal depth data generation: the minimal depth values are usually achieved on the ground, while the robot will never collide with the ground under our assumption. So we use the following (4) and (5) to filter out them. Furthermore, for depth data with excessive changes between adjacent points, we perform linear interpolation operations.

Without loss of generality, we assume the height of the camera from the ground is h; i.e., the height of the ground in the camera coordinate system is $-h$. Then, the height in the depth value of each pixel in M can be calculated using (3) ($z_c = d$ is the depth value of the pixel). We use a threshold ϵ to judge whether a pixel (u, v) in the map is near the ground:

$$|y_c + h| < \epsilon. \tag{4}$$

If a pixel (u, v) satisfies the above inequality, we set the depth $M(u, v) = 0$. Finally, we set the filtered 2D minimal depth data $L^{1\times W}$ as follows:

$$L(i) = \min M(j,i), j \in 1, \ldots, \frac{H}{2}, \text{where } M(i,j) \neq 0. \tag{5}$$

2D refers to the data obtained from the 2D LiDAR. As shown in the formula, what we finally get after fusion is actually a one-dimensional vector. Our generated 2D minimal depth data is then fused with the 2D LiDAR data. Specifically, for those 2D LiDAR data points which are covisible by 2D LiDAR and RGB camera, we replace their depth values with 2D minimal depth data. For those LiDAR data points which are not observed by RGB camera due to smaller FOV of RGB camera (62°) compared with the FOV of 2D LiDAR (180°), we maintain the original 2D LiDAR depth measurements.

C. Our DRL Module

In [16], the multi-robot collision avoidance problem without perfect sensing about the surrounding agents' positions, velocities, and shapes, is addressed by a reinforcement learning solution to a partially observable sequential decision making problem, which is formulated as a partially observable Markov decision process (POMDP). Proximal Policy Optimization (PPO) is used as policy gradient algorithm for multi-robot system and the Adam optimizer [38] is used to optimize the loss of PPO. Since the fused 2D depth data may still be partially accurate, we follow this solution and get a better result by replacing the multi-Stage training with our novel self-state-attention unit.

To find the best decision for robot navigation, we need to clarify the basis of the decision, the decision space, and the impact of the decision, which correspond to observation space, action space, and reward functions.

1) Observation Space: In this formulation, each robot only has access to the underlying LiDAR and camera observation information, self-motion state, and goal position. We define the observation space as $O = [o_l, o_s, o_g]$, where o_l represents the 2D LiDAR observation, o_s represents the current self motion state which includes linear and angular velocities, and o_g represents the Euclidean distance to the goal position.

2) Action Space: Action space consists of the linear velocity v with a range of $(-1, 1)$ and angular velocity ω with the angle range of $(-90°, +90°)$. For an agent, the action decision it makes at this moment will be the o_s in the next frame.

3) Reward Function: Our reward function is modeled as follows:

$$(r_1)_i^t = (r_d)_i^t + (r_c)_i^t + (r_\omega)_i^t \tag{6}$$

where

$$(r_c)_i^t = \begin{cases} r_{collision}, & \text{if } collision \\ r_{reach\_goal}, & \text{if } reach\ goal \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The reward $(r_1)_i^t$ for robot agent i at time step t is defined as the sum of Euclidean distance change $(r_d)_i^t$ (which is the difference of Euclidean distances to the goal before and after the current action), navigation result $(r_c)_i^t$ and rotation reward $(r_\omega)_i^t$. $(r_\omega)_i^t$ is a small penalty to punish the large angular velocities. r_{reach_goal} and $r_{collision}$ are constants which are set to reward or punish the navigation result.

This part we mainly follows the reward function proposed in [16], but we found that the weight and constant in their two-stage policy training reward functions are totally different. We try to use the same reward function for two stage training and get worse result. The difference between the two stage is that the number of agents increases, and the obstacles are from only dynamic obstacles to both static and dynamic obstacles. Because readjusting the weights of rewards for different environments is obviously laborious and prone to overfitting, we propose a concise and effective self-state-attention unit to replace multi-stage training and achieve better results.

4) Self-State-Attention Unit: We believe that the main reason for splitting into multi-stage training is that current feature is insufficient to overcome changes in the environment and the number of agents. These differences can be overcome if we can get more information from the input that is useful for policy training, and multi-stage training is no longer necessary. Since the input data is 2D observation, which represents the Euclidean distance between the closest obstacles and robot in a fixed range, the importance of the data point will be different according to robot type, goal position and velocity. For example, the robot we used in this letter will always turn to the forward direction which means the central part of 2D observation will be more important. Goal position and velocity will affect the importance of 2D observation in the similar way. Based on this, we introduce self-state-attention unit and utilize both 2D input and state attentions to augment the processed features, as shown in Fig. 2. F is the input depth data, and is concatenated with its goal and velocity vectors to regress the attention map. The attention map is then used to modulate the processed features to get the final remapped features R. C_h, C_I, C_g are three 1×1 conv.

IV. Experiments

In this section, we evaluate the proposed method in two parts. First, we evaluate the accuracy of reconstructed dense depth map

and the filtered 2D minimal depth data (Section IV-A). Then we evaluate the performance of our DRL framework in different test scenarios (Section IV-B). Finally, we demonstrate the real robot navigation by our whole system (Section IV-C).

A. Accuracy of Depth Estimation

In this section, we present two experiments. The first one is to compare our dense map reconstruction results with representative existing methods. The second one quantitatively compares the accuracy of filtered 2D minimal depth data.

1) Dataset: The NYU-Depth-v2 dataset [30] consists of RGB and depth images collected from 464 different indoor scenes, using a Microsoft Kinect device. We follow the official split to use 249 scenes for training and 215 scenes for testing. Test data includes 654 images with labels. The original size of NYU-Depth-v2 images is 640×480 and the input size is rescaled as 224×224 which is the same as the output size.

2) Training and Network: We train our model on the NYU-Depth-v2 dataset [30] using a NVIDIA TITAN X GPU. Our encoding layers consist of a ResNet and 3×3 convolution modules and decoding layers consist of 4 transpose convolution modules using 3×3 kernel. The weights of ResNet are pretrained on ImageNet [39]. The learning rate starts at 0.01 and is reduced by 20% every 5 epochs. We use a batch size of 8 and train for 15 epochs.

3) Metrics: We use root mean squared error (RMSE), mean absolute relative error (REL), and three types of percentages of predicted pixels where the relative error is within a threshold $(\delta_1, \delta_2, \delta_3)$ metrics to evaluate depth estimation:

$$RMSE = \sqrt{\frac{1}{N}\sum_i (\widehat{y}_i - y_i)^2} \tag{8}$$

$$REL = \frac{1}{N}\sum_i \frac{|\widehat{y}_i - y_i|}{y_i} \tag{9}$$

$$\delta_i = \frac{\mathbf{card}(\{\widehat{y}_i : max\{\frac{\widehat{y}_i}{y_i}, \frac{y_i}{\widehat{y}_i}\} < 1.25^i\})}{\mathbf{card}(\{y_i\})} \tag{10}$$

where y_i and $\widehat{y}_i$ are the ground truth and prediction, respectively, **card** is the set cardinality. To evaluate the accuracy of filtered 2D minimal depth data, we use RMSE, REL and mean area error (MAE). The MAE is calculated by the enclosed area between ground truth depth value curve and the evaluated depth value curve that can directly reflect the error. We perform interpolation to ensure the curves are enclosed completely.

4) Experimental Result: In Table I, we compare with representative methods on dense depth estimation. Two different types of methods are compared in this table: one is to estimate dense depths by using both RGB image data and 2D LiDAR sensor data, i.e., Liao *et al.* [36] and ours, and the other is to perform monocular depth estimation only on RGB images. The results show that using 2D LiDAR sensor data can significantly improve estimation results, and our method is better than Liao *et al.* [36] at four metrics REL, δ_1, δ_2 and δ_3. Considering that the collision avoidance task usually requires high accuracy about depth estimation, our method is more suitable for this task than existing methods. Some qualitative results are illustrated in Fig. 3.

TABLE I
COMPARISON OF DEPTH ESTIMATION

Method	RMSE	REL	δ_1	δ_2	δ_3
Fast depth [40]	0.604	0.165	77.1	93.7	98.0
Eigen et la. [28]	0.907	0.215	61.1	88.7	97.1
Liu et al. [41]	0.824	0.230	61.4	88.3	97.1
Cao et al. [42]	0.645	0.150	76.9	95.0	98.8
Laina et al. [29]	0.583	0.129	80.1	95.0	98.9
Sparse to Dense [32]	0.555	0.154	79.3	95.1	98.6
Liao et al. [36]	**0.442**	0.104	87.8	96.4	98.9
Ours	0.455	**0.102**	**89.0**	**97.4**	**99.3**

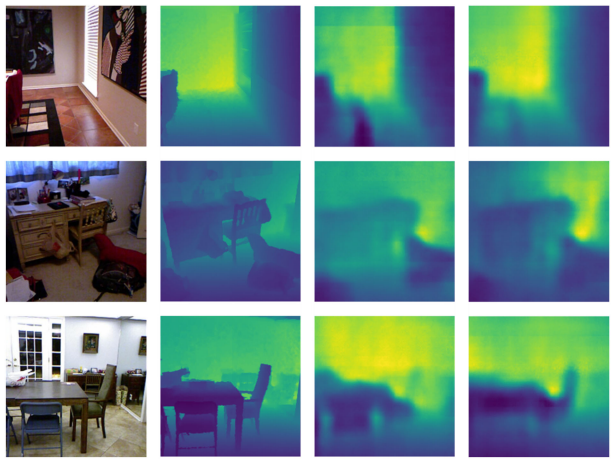

Fig. 3. The example of depth estimation comparison. The 4 columns of images from left to right are original image, ground truth of depth image, depth estimation of Sparse to Dense [32] and our result. Obviously our depth estimate is closer to ground truth.

TABLE II
COMPARISON OF ERRORS WITH RESPECT TO GROUND TRUTH

	MAE	RMSE	REL
Original 2D LiDAR sensor data	49.8	55.5	1.33
Filtered 2D minimal depth data	**7.42**	**10.4**	**0.168**

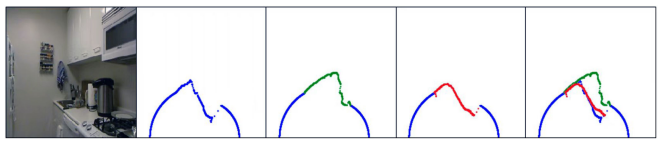

Fig. 4. From left to right: the RGB image, the ground truth for fused depth shown in blue color, 2D LiDAR sensor data (used as a rough approximation of 2D minimal depth data for collision avoidance) shown in green color, filtered 2D minimal depth data shown in red color, and the overlap of three data.

Table II shows the comparison of errors with respect to ground truth, between original 2D LiDAR sensor data (used as a rough approximation of 2D minimal depth data for collision avoidance) and our filtered 2D minimal depth data. The results indicate that the errors are significantly reduced in our filtered 2D data, which can help robot take more reasonable actions. A graphical example is illustrated in Fig. 4.

Both experiments on dense depth estimation and filtered 2D minimal depth data demonstrate the effectiveness of fusing

2D LiDAR sensor data and monocular RGB images in robot navigation and collision avoidance.

B. The Performance of DRL Framework

We test our DRL framework and compare its performance in different test scenarios. Due to the lack of a unified training and testing environment, the comparative experiments of multi-agent DRL are often not convincing enough. Therefore, we choose the widely recognized work that performs best in many cases as our baseline, add our self-state-attention unit and use only one-stage training. For training, we have used their scene of stage 2, and for test, we have designed four new test scenes ranging from simple to complex because the original test scene is a bit homogenous. With the model parameters consistent with their original stage2, we added our improvements and did comparative experiments to show that our method are concise and effective.

1) Parameter Details: For reward function, $r_{collision}$ and r_{reach_goal} are set to -10 and $+10$. k is set to 3. Some main hyper-parameters of training, the training episode, batch size, max timestep, learning rate, KL penalty and discount factor are set to 15000, 512, 256, $5e^{-5}$, $15e^{-4}$, 0.99, respectively.

2) Implementation Details: Our DRL framework is trained using Pytorch, deployed on a PC with Intel E5-2600-v3 CPU, 256 GB RAM, and NVIDIA RTX2080Ti. All the training and testing experiments are based on Stage mobile simulator, including the scenarios definition, multi-robots definition, movement simulation, and data generation. During training and testing, we only need to simulate 2D LiDAR sensor data and the robot's movement in the environment. The whole training process includes 15,000 epochs.

3) Evaluation Metrics: We choose the two widely used metrics, i.e., success rate and average time step, to verify the effectiveness of our DRL framework. Success rate represents the proportion of samples that successfully reach the goal position within the max time step. The average time step is obtained by the average of the last time steps of all success samples.

4) Scenarios Definition: Our training scenarios are mainly based on [16] with some fine-tuning. For the Partially Observable Markov Decision Process (POMDP) we use in Section III-C, each moving robot in the same scenario is a dynamic obstacle for other robots and then the scheme trained in the multi-robot scenarios can also be used in a real world with multiple dynamic obstacles. To evaluate our DRL framework in different scenarios, we design (1) a static scenario for one moving robot, (2) two dynamic scenarios with moving robots, and (3) a random complex obstacle scenario with multiple moving robots to test the generalization and effectiveness of our framework. All four test scenarios are summarized in Fig. 5. The goal position of each robot is mirrored and symmetrical with the initial position about the center of the scenarios.

5) Reward Comparison: The reward function we used for our method is exactly the same as the Stage2 of [16], which provides the basis for fair comparison. We train the stage2 only, stage2 after stage1 of [16] and our method at the same PC for 3 times, select the best one and draw the reward curve.

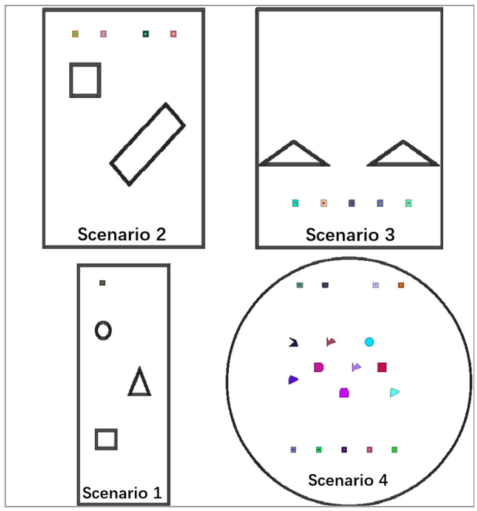

Fig. 5. Four test scenarios used in Section IV-B. The rectangle with colored center are robots. Irregular colored objects in scenario 4 are randomly generated obstacles. Note that the position of obstacles in scenario 4 are changing with robot reset. The goal position of each robot is mirrored and symmetrical with the initial position about the center of the scenarios.

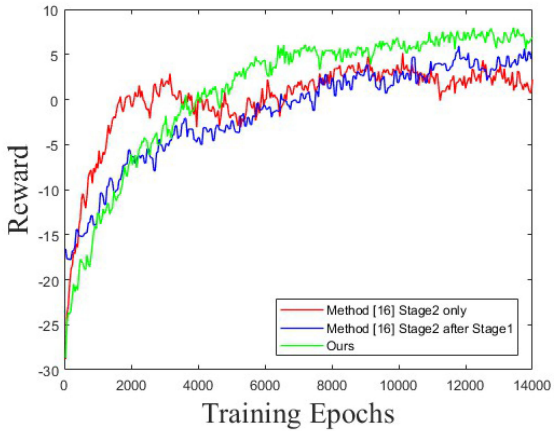

Fig. 6. Reward comparison for the Stage2 only policy and Stage2 after Stage1 policy of method [16] and ours.

As Fig. 6 shown, from the 4000 epoch, the reward of our method keeps the best and finally converges to a higher value. The average reward of our method, stage2 only and stage2 after stage1 of method [16] during the their best 1000 epochs are 6.3, 4.5 and 3.1, which shows that our method converges to better performance in training scenarios.

6) Evaluation Results: As shown in Table III, our method achieves both the best average success rate and average time. From the results, we can draw some conclusions. Firstly, simple static scenario for one moving robot is easy to reach the goal position. Secondly, all methods can handle dynamic environments of similar complexity to training scenarios with moving robots relatively well, and the method [16] get the best performance. Thirdly, when the scenario becomes more random, complicated and unfamiliar complexity, the effect of method [16] decreases significantly, but our method has a good generalization and

TABLE III
COMPARISON RESULTS OF OUR DRL SCHEME AND THE METHOD [16] ON FOUR TEST SCENARIOS. SCE IS SHORT FOR SCENARIO. THE UNIT OF TIME IS SECONDS

	metrics	sce 1	sce 2,3	sce 4	average
Stage2 only [16]	success rate	**1**	0.768	0.489	0.648
	average time	13.02	**11.66**	16.18	14.18
Method [16]	success rate	**1**	**0.841**	0.403	0.661
	average time	14.77	13.93	16.11	15.07
Ours	success rate	**1**	0.774	**0.750**	**0.775**
	average time	**13.00**	12.37	**14.48**	**13.38**

Fig. 7. Snapshots of real robot navigation with our novel DRL framework. Note that the drawers at high positions serve as obstacles that do not exist near the ground.

still maintains high efficiency. Since the weight of the reward function in [16] is adjusted with the change of the complexity of the training environment, we believe the high performance for method [16] in sce 2 and 3 may be due to some overfitting caused by multi-stage training, which can also be proved by the result that Stage2 only [16] is better than method [16] in sce 4.

C. Real Robot Navigation

As shown in Fig. 1, our multi-functional robot has a 2D LiDAR sensor in the middle and a camera on the top. A laptop with AMD Ryzen 7 4800H CPU, 32 GB RAM and NVIDIA RTX1060Ti is used to perform data processing and action prediction.

During the navigation task, the robot receives RGB images from the camera, sparse point cloud information (all points have the same height) from the 2D LiDAR sensor, and position information by the location sensor in the robot. The velocity information is computed using the previous time step. Our pipeline is performed at 10 Hz and each action is determined by our proposed method that utilizes dense depth estimation, 2D filtered minimal depth data generation, fusion with 2D LiDAR data and a novel DRL framework. In the real world experiment, the nearby obstacles are sometimes located in the blind spot of the camera. The obstacle distance we get through the RGB image will be updated and maintained for two timesteps by sensing the movement distance of the robot.

Snapshots of real-world robot navigation using our novel method are shown in Fig. 7. We use movable cabinets to work as obstacles, and open drawers are high obstacles that are invisible to 2D LiDAR sensor. In this scenario, the collision rate of the LiDAR-based method is almost 100%, while the chance of our pipeline passing without collision can be close to 80%. Examples can be found in the accompanying demo video.

V. CONCLUSION

In this letter, we propose a novel self-state-attention DRL scheme and a simple yet effective navigation method for high robots. A 2D minimal depth data generation and fusion method is proposed to utilize both the 2D LiDAR data and RGB data to adapt to high robots navigation. A novel DRL training framework is developed by taking the relation between robot motion state and perception into account. Experiments show our proposed method's effectiveness of fusing 2D LiDAR data and monocular RGB images and improved performance of our DRL policy on success rate, average time and generalization capability. Furthermore, our method has been applied to a real robot and demonstrates its effectiveness in real scenarios.

REFERENCES

[1] F. A. Cosío and M. P. Castaneda, "Autonomous robot navigation using adaptive potential fields," *Math. Comput. Modelling*, vol. 40, no. 9/10, pp. 1141–1156, 2004.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Automat. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[3] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1990, pp. 572–577.

[4] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Automat.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.

[5] J. Van den BergM. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 1928–1935.

[6] J. Van Den, S. J. Berg, M. Guy Lin, and D. Manocha, "Reciprocal N-body collision avoidance," in *Robotics Research*. Berlin, Germany: Springer, 2011, pp. 3–19.

[7] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011.

[8] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "Smooth and collision-free navigation for multiple robots under differential-drive constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 4584–4589.

[9] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed Autonomous Robotic Systems*. Berlin, Germany: Springer, 2013, pp. 203–216.

[10] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1343–1350.

[11] T. Fan, X. Cheng, J. Pan, D. Manocha, and R. Yang, "Crowd-move: Autonomous mapless navigation in crowded scenarios," 2018, *arXiv:1807.07870*.

[12] A. Faust *et al.*, "PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5113–5120.

[13] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 593–600.

[14] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," in *Proc. Int. Conf. Learn. Representations*, 2016.

[15] J. Choi, K. Park, M. Kim, and S. Seok, "Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 5993–6000.

[16] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 856–892, 2020.

[17] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6252–6259.

[18] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2759–2764.
[19] L. Ma, Y. Liu, J. Chen, and D. Jin, "Learning to navigate in indoor environments: From memorizing to reasoning," 2019, *arXiv:1904.06933*.
[20] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull, "A data-efficient framework for training and sim-to-real transfer of navigation policies," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 782–788.
[21] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 31–36.
[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[23] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5129–5136.
[24] H. Zeng, R. Hu, X. Huang, and Z. Peng, "Robot navigation in crowd based on dual social attention deep reinforcement learning," *Math. Problems Eng.*, vol. 2021, pp. 1–11, 2021.
[25] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5671–5677.
[26] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6015–6022.
[27] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 797–803.
[28] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.
[29] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proc. 4th Int. Conf. 3D Vis.*, 2016, pp. 239–248.
[30] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.
[31] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, May 2009.
[32] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4796–4803.
[33] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, "Sparse and noisy LiDAR completion with RGB guidance and uncertainty," in *Proc. IEEE 16th Int. Conf. Mach. Vis. Appl.*, 2019, pp. 1–6.
[34] J. Tang, F.-P. Tian, W. Feng, J. Li, and P. Tan, "Learning guided convolutional network for depth completion," *IEEE Trans. Image Process.*, vol. 30, pp. 1116–1129, 2021.
[35] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[36] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, and Y. Liu, "Parse geometry from a line: Monocular depth estimation with partial laser observation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5059–5066.
[37] H. Lim, H. Gil, and H. Myung, "MSDPN: Monocular depth prediction with partial laser observation using multi-stage neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020, pp. 10750–10757.
[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[39] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[40] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "FastDepth: Fast monocular depth estimation on embedded systems," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6101–6108.
[41] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, Oct. 2016.
[42] Y. Cao, Z. Wu, and C. Shen, "Estimating depth from monocular images as classification using deep fully convolutional residual networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 11, pp. 3174–3182, Nov. 2018.