A LiDAR based end to end controller for robot navigation using deep neural network

J.K. Wang, X.Q. Ding, H. Xia, Y. Wang, L. Tang, R. Xiong

Abstract—Navigation is a primary task for mobile robot to accomplish its tasks. Conventionally, these navigation methods require lots of parameters to be tuned artificially. In this paper, we propose a navigation method to learn the end-to-end control policy using convolution neural network, which directly outputs the velocity and angular rates using only the current observation and the target. Besides, a sliding window of past information is incorporated to add the memory to the controller, so that the hesitation is reduced when ambiguity occurs. To train the model, the data is generated from expert planner in the simulation environment, leading to a low cost for massive data. To validate the trained end-to-end controller, we compare our method with the expert global planner both in the training map and complex map in the simulation environment.

Keywords—navigation, deep neural network, end-to-end

I. INTRODUCTION

Path planning and navigation have long been a popular and classical topic in field of robotics. This problem is described as guiding robot from the current point to a given target point and keeping a safe distance to obstacles. Traditional methods often require comprehensive data processing in order to achieve this goal, in which many parameters need to be tuned manually. In addition, to achieve the path planning, a map with current obstacles information is mostly required, which makes navigation in a strange environment difficult.

In this work, we propose a navigation method using deep neural network to reduce the difficulty of parameter tuning. By learning control commands based on a global path planner, our model generates an end-to-end controller with current observation and target as input and linear and angular velocity as output. In order to keep the continuity of controller, a sliding window is used to add the memory to the controller. During the test in the simulation environment, our method can not only navigate safely in the training map but also in the unseen map.

Compared with the traditional method, the deep neural network has a unique ability to understand the complex environment. Our network consists of two parts. A convolution neural networks are used to process lidar observation in the first part, which can reduce the amount of network parameters. And then the target information and CNN output are merged and feed into the fully-connected layers, which output control commands. Deep neural network requires extremely large amounts of data as training samples, and it is not easy to collect such large-scale data in real-world scenarios. So, we chose to build a training map in the simulation environment, and by continually generating random target points, we obtained 140,000 frames of training samples generated by the global path planner. Since our training set data is the velocity and angular rates of the car, the output of our model can be used directly as a car control command without the need for modeling.

In order to test the stability of our method, we test our model both in training map and unseen map. We compare our method with global path planner in two ways. One is to compare the output of two methods frame by frame while driving on a fixed path and the other is to choose the target points randomly and compare the path generated by two methods.

In Section 2 we present an overview of the related work. Section 3 presents a model for the problem and introduces our approach. The performances of our work are shown in Section 4 and we discuss our result in Section 5. Finally, we draw a conclusion in Section 6.

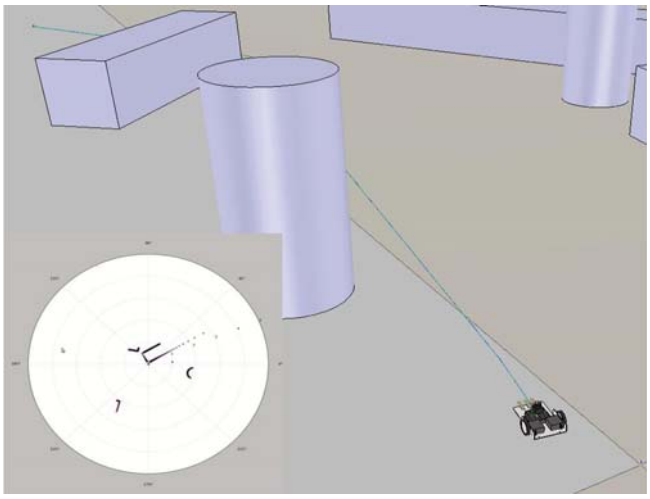

Fig. 1: The vehicle can pass through the obstacles and reach the target point accurately. The radar chart shows the obstacles information observed by the vehicle lidar.

II. RELATED WORKS

Our work is closely related to many areas of robotics, such as perception and motion planning. So, we will introduce the related works in two parts.

Perception requires us to extract useful information from the data observed by the sensor. Traditional methods usually extract the features from data and then try to understand the features. C. Farabet *et al.* [1] showed that it's possible to build an abstract model of the environment shown in images. A method proposed by Chen *et al*. [2] not only extracted the features from the image data, but also applied it to the automatic driving field.

However, in most cases the image method is less accurate than the laser method. So, our approach is more suitable for working in complex environment. Ondurska *et al*. [3] presented an end-to-end application of neural network for dynamic object tracking using laser data. Their work proves that 2D laser can

978-1-5386-3107-2/17/$31.00 ©2017 IEEE

characterize spatial information and only depends on very small amount of data compared with multidimensional laser.

The motion planner needs to process the information perceived and make motion decisions. Abbeel *et al.* [4] trained a path planner for parking lots by learning from human parking behavior. However, their method requires prior knowledge of obstacle information, which leads to the inability in unknown environment. Levine *et al.* [5] showed that end-to-end learning approaches can be used in robotic arm control. They trained a convolutional neural network to plan motor torques for a robotic arm, given raw image data.

In the field of mobile robots, Rose *et al.* [6] presented an approach that can control the UAV left and right by learning from the raw image information. And Kim *et al.* [7] extended above approach by learning translational and rotational velocities. They tested their approach in empty hallways without obstacles.

The use of laser data for motion planning is also a current research hotspot. Mark *et al.* [8] presented a model that is able to learn the complex mapping from raw 2D-laser range findings and a target positing to the required steering commands for the robot. But the output of their network is only determined by the current frame observation, which may cause the vehicle to fall into an infinite loop.

III. METHOD

In this section, we present following works. First of all, we model the problem and put forward our approach to solve it. Then we show the composition of our deep neural network and explain its working principle. After that, we explain the method of obtaining training data and the details of the training. Finally, we improve the previous network to make it have a better performance.

A. Modeling

When driving a car, humans can process the input observations in the brain and output the acceleration and turning commands directly. Inspired by the human behavior, we model this problem as follows:

Assume that the only information we know is the lidar's observation **L** and the relative position of target point **T**. The output **u** that we need to control the vehicle consists of velocity **v** and angular rate **w.** We need to find the appropriate function $\boldsymbol{F_\theta}$ to meet

$$u = F_\theta(L, T)$$

in which θ is the parameters to be learned from the training set. To find the function $\boldsymbol{F_\theta}$, we have to minimize the loss function

$$L = \frac{1}{2}(F_\theta(L, T) - u_{exp})^2$$

in which u_{exp} is the expert output **v** and **w** produced by a global path planner. And we will explain how to generate this information from the global path planner in part C.

B. Network

Deep neural network has a strong ability to model the multi-parameters, strongly coupled systems. Because our vehicle is equipped with a lidar with resolution of 0.25 degrees and observation range of 270 degrees, our laser data has 1080 data points per frame. If the data is feed to the fully-connected network directly, there will be too many parameters to learn, resulting in very slow training convergence speed.

In order to increase the depth of the network at the same time avoiding excessive parameters, we firstly process the laser input with a five-layer convolution neural network. Then we reshape the output of the CNN and connect it with the target data as the input of the fully-connected layers.

In addition, we use batch normalization after each convolution layer to speed up the training. Besides, Residual networks are added in CNN to prevent the gradient dispersion. At the same time, we find that the dimensions of target data are too small compared to the laser data. So, we need to raise the dimensions of the target data. In the first version of our work, we constantly copy and paste the target data until it reaches the desired dimensions. In the second version, we process the target data with three deconvolution layers to raise dimensions and we get same performance as version 1 with less final dimensions. Our model implementation is based on TensorFlow framework [9]. More details about our model are shown in Figure 2.

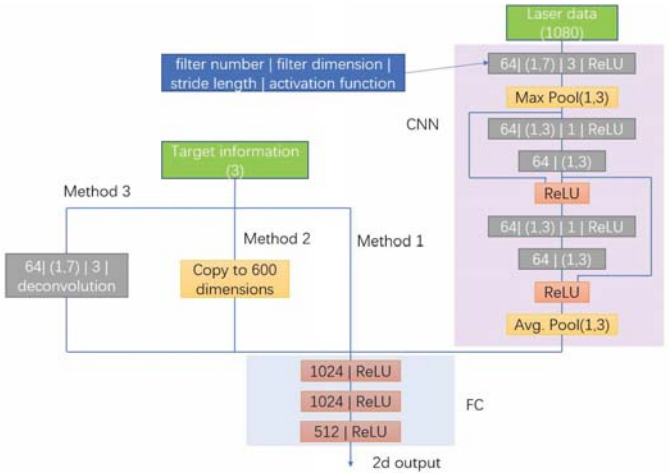

Fig. 2: Structure of our network. The laser data is processed by the CNN first and two residual building blocks are added into the CNN. Three methods are tried in our work to raise the dimensions of the target information before connecting with the CNN output. The fully-connected network consists of three layers and output a 2D control command finally

C. Training

In order to train our model, we first need to collect a lot of data from simulation environment. In our work, we choose V-REP robot simulator [10] to generate our training data. As the function shown in part A, the training data should consist of laser measurements, target information and control command. To generate the control command, we use a global path planner to calculate a safe path and a simple PID controller to make sure the vehicle move on the path.

With enough data, we start to feed it into network in batches to train our model. For each iteration, we randomly input 32 tuples which consist of lidar's observation **L**, target information **T** and the control command $\boldsymbol{u_{exp}}$ into network. So, for learning step k, the loss function can be expressed as:

$$J_k(D_i) = \frac{1}{32}\sum_{j=1}^{3\,2}(\,F_{\theta_k}(\,L_k\,,\,T_k\,) - \,u_{exp,k})^2$$

where D_i is a random collection of the 32 tuples we mentioned above and θ_k is the model parameters in step k. The Adam optimizer [11] is used as the optimization while training.

D. Improvement

With the model in Part B, our vehicle can run safely in most cases. However, in some special situations, control hesitation will waste a lot of time. Figure 3 shows an example of the special condition when our vehicle is faced with a long straight obstacle and the target point is on the other side of the obstacle. When the vehicle firstly turns left, the lidar will see a long obstacle. This may make the network outputting a right turn command later, with which the lidar will observe the obstacle too, falling into a loop.

We solve this problem by introducing memory information into the model. The target information in Part B is replaced by a combination of the last three frames of target information.as shown below. This change greatly reduces the time of hesitation and the control curve also becomes much smoother than before.

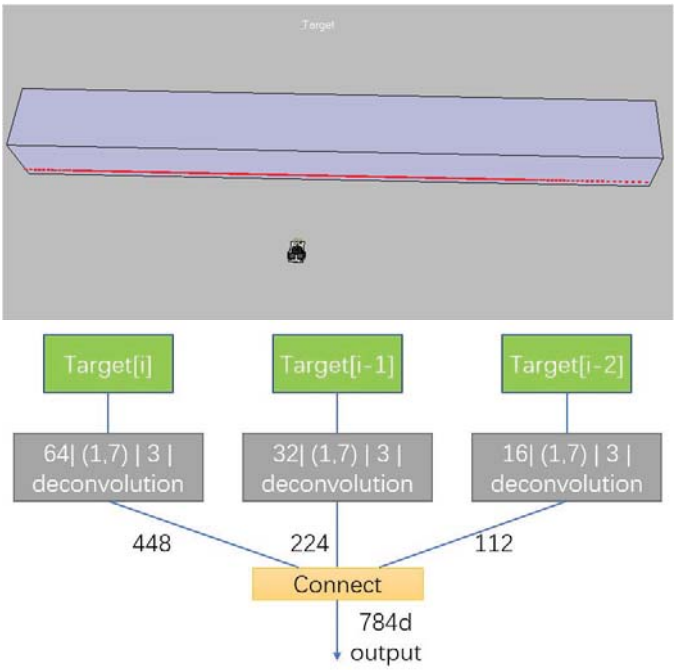

Fig. 3: When the vehicle is in the environment shown above, the output of the previous network will be shocked. The vehicle observed the right obstacles and turned left, however turning left will meet the same problem. And the vehicle will eventually fall into an endless cycle. The bottom figure shows the target input part of our network that improved with memory information.

IV. Experiments

In this section, we present the experiments and their evaluation. First, we introduce our system configuration, simulation environment and training data. Next, we compare the performance of our model with the global path planner frame by frame. Then our model is used to navigate the vehicle through the obstacles both in training map and unseen map. At last we show the performance after adding the improvement of our model.

A. Environment and configuration

All of our work is running on Ubuntu 14.04 and we simulate on V-REP robot simulator 3.3.1 EDU version. Robot Operating System (ROS) [12] is also used as a bridge to collect training data from V-REP and pass control commands into the simulation environment based on network output. We build a two-wheel drive car equipped with a lidar in simulation environment. The model of the lidar is UTM-30LX with angle resolution of 0.25 degrees and measuring range of 270 degrees.

In order to obtain training data, we use the V-REP built-in path planner to generate the trajectory first, and then use a PID controller to control the vehicle travel along the path. So, we can obtain the translation and rotation value directly without modeling the vehicle.

After more than 1000 path traversal, we obtained 140,000 frames of training data with size of 1.6 GB. Training the model on a Nvidia GeForce GTX 980 Ti GPU takes 4 h with 200,000 training steps.

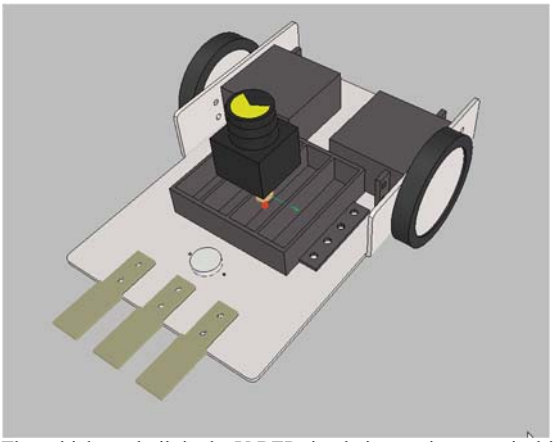

Fig. 4: The vehicle we built in the V-REP simulation environment is driven by two rear wheel differentials, equipped with a 2D-lidar to sense obstacle information. A PID controller is used when collecting data and our model is used when test.

B. Frame-by-frame

In this experiment, we use the global path planner to generate the path to target points chosen randomly. The vehicle collects the lidar data which is entered into our model while moving on the path. The output of our model is compared with the output of the PID controller under the global path planner. In fact, this work plays a same role as the test set, which can test the difference between our model and the expert controller. Although our network is trained on the GPU, once the training is done, the network parameters are fixed and the network can be run on CPU. Our model can achieve a processing speed of 60 frames per second, which can completely meet the real-time requirements. We tested the three target input types shown in Figure 2 both on the training map and the unseen map. The results are shown in Figure 5.

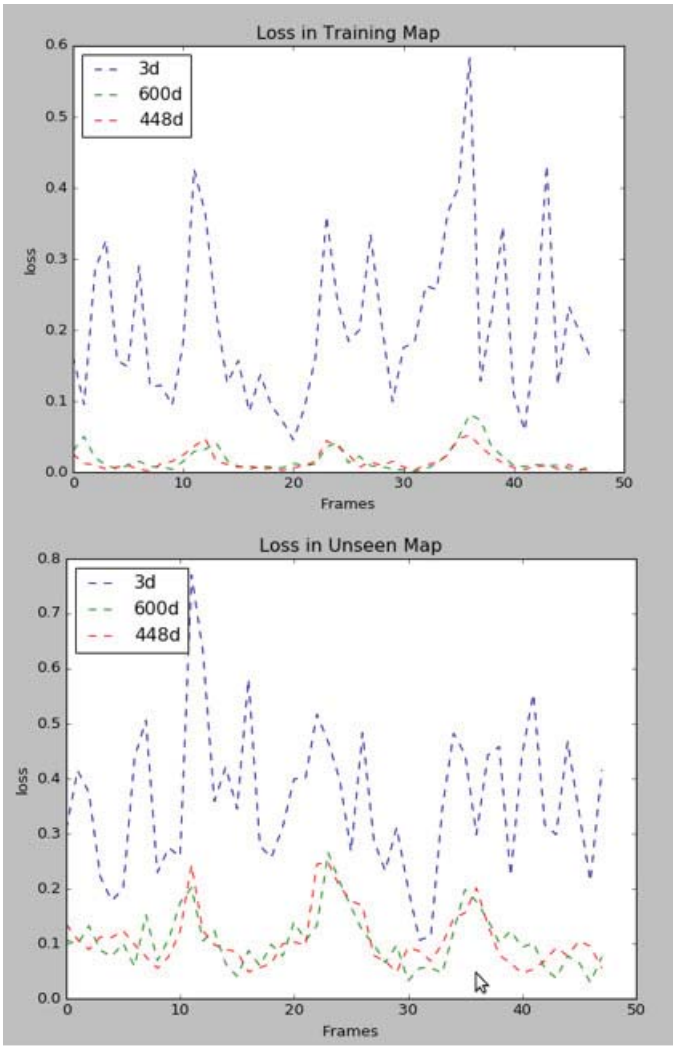

Fig. 5: The error between our model and the expert planner. Blue line represents the original target(3d) input into FC layers directly. The green and red lines respectively show the copy method and deconvolution method. We average every ten frames into one frame to prevent data jitter.

From the experiment results we can conclude that inputting the 3d-target information directly into FC layers works worst. The other two methods are not comparable in both maps, and they all work better in training map, which is easy to understand. We also found that the error occurs mainly at the car turns, but it doesn't mean that our model is weak at rotating. For example, the choice about turning time and counterclockwise will increase error significantly, but the actual impact is not as big as it looks. Therefore, testing in a fully autonomous situation is necessary.

C. *The whole path*

Previous experiments have shown that, for isolated frames, the output of our model can be similar with the expert planner. In order to verify whether our model can navigate autonomously, in this experiment, we use our model to control the vehicle independently.

We also carry out experiments in training map and unseen map. The vehicle starts at the starting point and moves toward the randomly generated target point. When the target point is reached, the other target point is recreated immediately. A continuous arrival of 6 target points is considered a successful round. We compare the above three methods from three aspects of average running time, average energy consumption and running success rate.

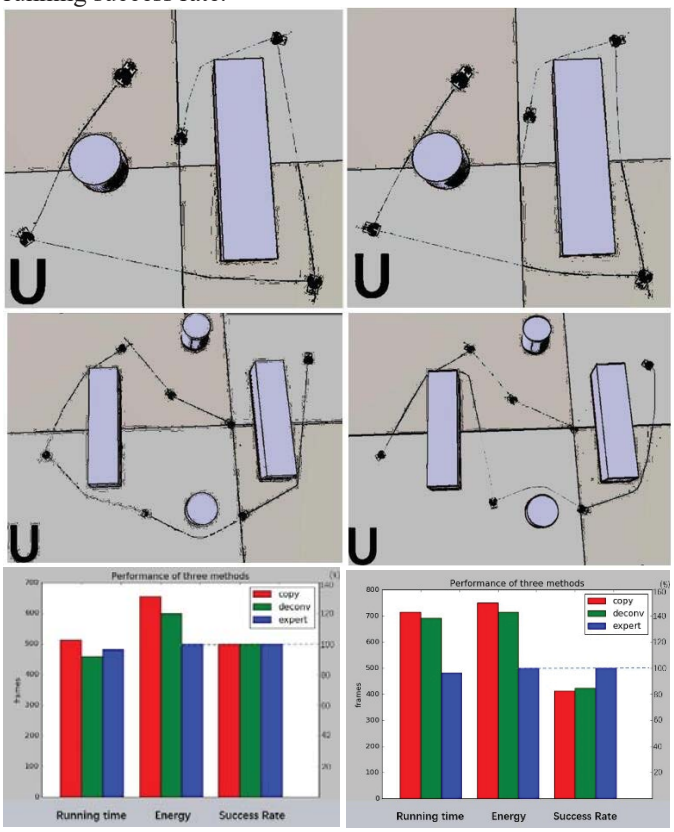

Fig. 6: The two pictures of the first row show the comparison of the running trajectory between our model (deconvolution method) and the expert planner in training map. The expert result is shown on the left and our model is on the right. Because the copy method is similar to deconvolution method in most condition, we only show the comparison of the deconvolution method and expert method. The second-row pictures show comparisons in unseen map. The bar chart shows the average performance of copy, deconvolution and expert method after 30 rounds of testing in training map(left) and unseen map(right).

From this experiment, we can conclude that in the training map, the path generated by the latter two methods is almost the same as that of the expert planner. The method of directly entering target information into the network never run successfully, so we did not compare it in the diagram.

The performance of other two methods can even exceed the expert planner in the training map, and our models can reach the target point more quickly. In the unseen map, the performance of our models declines a little but still performs well in most tasks even the time spent has risen by a big margin.

D. *Improvement*

In the last experiment, we found that a lot of time is spent on the clockwise and counterclockwise choice of the turn. The vehicle will hesitate and try again and again. To overcome this problem, we modify the target input as shown in Figure 3. After modification, historical information will also influence current decisions, and we hope to reduce the hesitation of our model with this method.

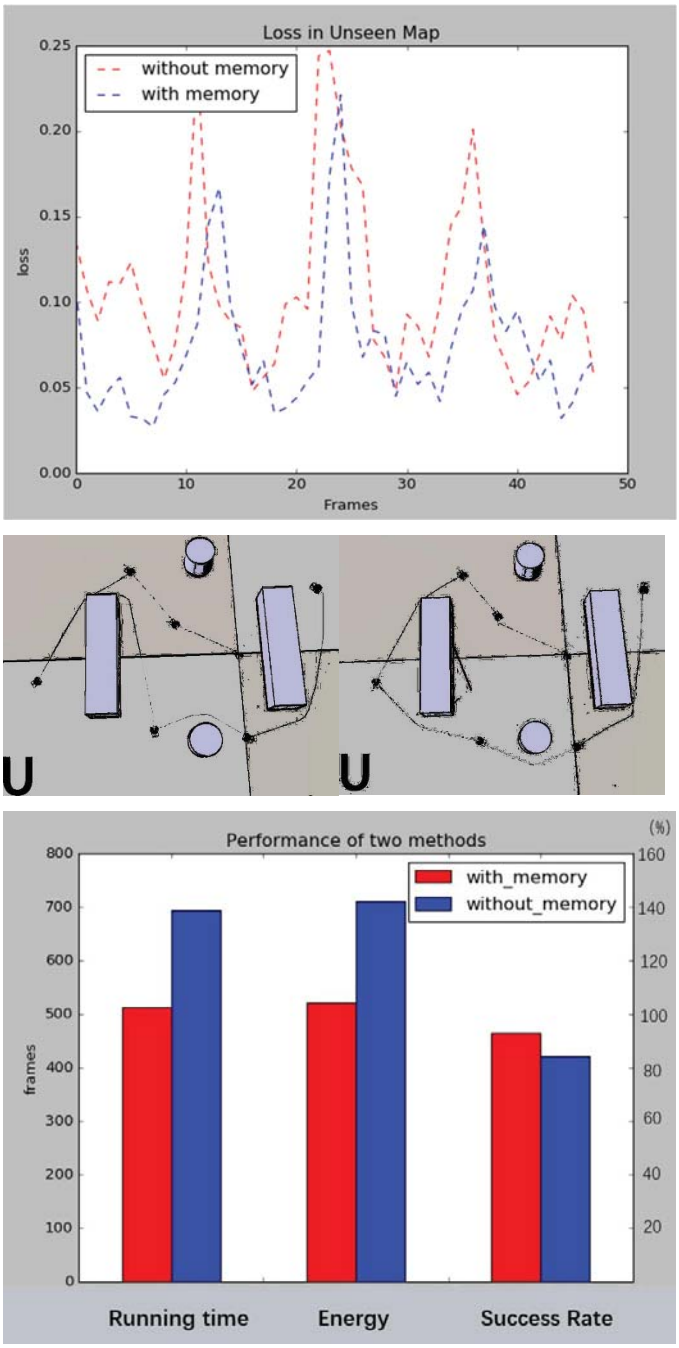

Fig. 7: The line chart compares the loss frame by frame between the methods before and after introduction memory information. The left picture of second-row shows the trajectory generated without memory while the right shows the trajectory with memory. The bar chart shows the average performance of target with memory and target without memory after 30 rounds of testing.

We first compare the difference between the deconvolution method with memory and without memory. Then we let them run autonomously between random targets and compare the paths they generate, as well as the time and energy they consume.

As we can see from Figure 7, when the memory information is added, the frame error is decreased, and the driving time and energy consumption are decreased to seventy percent compared with no memory information and the success rate is also increased. This shows that introducing prior frames information can reduce control hesitation and enhance strategy continuity.

V. DISCUSSION

In this work, we propose an end to end controller for robot navigation using deep neural network. Our method has the following advantages: (i) Our model is end-to-end, reducing a lot of parameters to be adjusted. (ii) Data acquisition occurs in a simulation environment where data is very easy to collect. (iii) Once the training is complete, the network parameters will remain fixed, so no matter how complicated the environment, our model can always give out the output in a very short period.

We demonstrate that our method is effective by comparing our method with global path planner. In frame-by-frame comparison, the loss that our model compared with global path planner is very small in training map. In unseen map, the error has increased but not more than 30% of the expert output. What's more, we find that the error occurs mainly at the beginning of the turn moment, which is caused by the different choice of turning time, but this error will not cause a great impact in actual test.

As mentioned above, in some cases the big error does not mean that our model is poorly performing, so we randomly generate targets for two planners and let them drive to the targets individually. We compare two planners in many aspects and find that our model even performer better than global path planner in training map. As the map becomes more complex, the performance of our model in unseen map has declined but it can still complete most of the missions. After that, we improve our model with memory information and make great progress in terms of time.

But our method also has some flaws. When we set the end target, we assume that the target point is not inside the obstacles and the distance between current position and target point needs to be known. But these problems can be solved in practical application. For example, we can expand the size of target range in case that the target point is unreachable and an odometer can be used to estimate the relative distance to the target point.

VI. CONCLUSION

We presented an end-to-end motion planning approach. It can learn the end-to-end control policy using convolution neural network, which directly outputs the velocity and angular rates using only the current observation and the target. And we improved our work with memory information and achieved good results.

REFERENCES

[1] C. Farabet, et al., "Learning hierarchical features for scene labeling," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1915–1929, 2013.

[2] C. Chen, et al., "Deepdriving: Learning affordance for direct perception in autonomous driving," in Proc. of the IEEE Int. Conf. on Computer Vision (ICCV), 2015, pp. 2722–2730.

[3] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[4] P. Abbeel, et al., “Apprenticeship learning for motion planning with application to parking lot navigation,” in Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Nice, France, Sept. 2008, pp. 1083–1090.

[5] S. Levine, et al., “End-to-end training of deep visuomotor policies,” Journal of Machine Learning Research, vol. 17, no. 39, pp. 1–40,2016.

[6] S. Ross, et al., “Learning monocular reactive uav control in cluttered natural environments,” in Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013, pp. 1765–1772.

[7] D. K. Kim and T. Chen, “Deep neural network for real-time autonomous indoor navigation,” arXiv preprint arXiv:1511.04668, 2015.

[8] M. Pfeiffer et al. “From perception to decision: a data-driven approach to end-to-end Motion planning for autonomous ground robots” arXiv preprint arXiv:1609.07910v1, 2016

[9] M. Abadi, et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[10] Rohmer E, Singh S P N, Freese M. V-REP: A versatile and scalable robot simulation framework[C]// Ieee/rsj International Conference on Intelligent Robots and Systems. IEEE, 2013:1321-1326.

[11] J. Ba and D. Kingma, “Adam: A method for stochastic optimization,” in In Proc. of Int. Conf. on Learning Representations (ICLR), 2015.

[12] M. Quigley, et al., “Ros: an open-source robot operating system,” in ICRA workshop on open source software. Kobe, Japan, 2009, p. 5.