2018 International Conference on Future Technology and Disruptive Innovation
Hangzhou, China, December 7-8, 2018

Learning Continuous Control through Proximal Policy Optimization for Mobile Robot Navigation

Taiping Zeng[1,2,3]
[1]*State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China*
[2]*Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110016, China*
[3]*University of Chinese Academy of Sciences, Beijing 100049, China*
zengtaiping.ac@gmail.com

Abstract

An intelligent mobile robot must be able to autonomously navigate in complex environments, so that it could be deployed in the real world. Traditional methods solve this problem by building a map of an environment, locating the position of the robot, and performing path planning to navigate the robot on the map. However, these methods often make a variety of assumptions and require intensive computational resources, which may restrict the application of these methods. More importantly, these methods lack of mechanisms to learn from failures. In this paper, I present a learning-based mapless mobile robot navigation method with continuous state and action spaces, in which a proved efficient policy gradient method, i.e. Proximal Policy Optimization (PPO), is introduced for learning continuous control tasks. It takes the normalized laser scanning data as input and directly outputs the continuous velocity commands to direct a mobile robot operating in the environments. The proposed method is trained end-to-end in several simulation environments to evaluate the performance without any manually designed features, human-provided labels, or prior assumptions. Experimental results show that it can learn to navigate through multiple different environments with a few hours of fully autonomous training. Also, it successfully learned to provide continuous control commands for mobile robots. Moreover, evaluations in multiple complex environments demonstrate the robustness and adaptability of the proposed method. The proposed learning-based method and mobile robot learning system can be a general approach to train mobile robots for more complex continuous tasks. Videos of the experiments can be found at https://youtu.be/P0bwzXI4EEA.

Index Terms

Mobile Robots, Deep Reinforcement Learning, Continuous Control, Proximal Policy Optimization, Robot Navigation, Mobile Robot Learning

I. Introduction

The ability to autonomously navigate through complex and unstructured environments plays an important role in mobile robot navigation, which ensures mobile robots can safely and efficiently work in a variety of challenging practical applications, such as delivery, search and rescue, inspection, and transportation. The main goal of mobile robot navigation is to determine its own position and then plan a path towards the goal locations. In order to navigate through environments, mobile robots are required to build and interpret the map of environments. As a result, mobile robot navigation can be defined as the combination of map building and interpretation, self-localization, and path planning to navigate mobile robots through environments without obstacle collision [1].

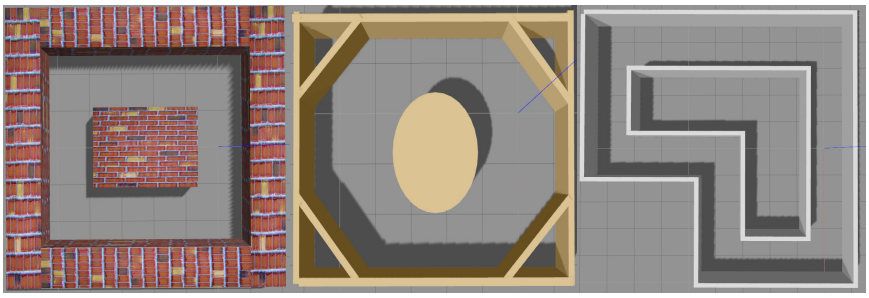

Fig. 1. Bird's-eye view of the virtual training environments simulated by the Gazebo simulator. The left, the middle, and the right shows the square-shaped environment, the round-shaped environment, the L-shaped environment, respectively.

Autonomous robot navigation has been extensively investigated, ranging from small size indoor mobile robots [2], [3] to large size autonomous driving vehicles in urban environments [4], [5]. Many of these traditional works need to build a map of the environments when localizing and planning movement control actions. Although geometric reconstruction and mapping methods have made a great progress in a range of navigation and object avoidance [6], it is still constrained by the size, weight, and power of mobile robots [7]. Considerable computational resources are needed for explicit modeling of the environment. A variety of assumptions are included, such as static or texture environments, which may not be suitable for practical scenarios. However, these limitations may lead to difficulties with textureless environments, dynamic tasks, and high-bandwidth sensors [8].

Learning-based methods attempt to address mobile robot navigation problems by learning from data, which provides considerable promise. Supervised learning methods can learn drivable routes [9], obstacle detectors [10], end-to-end driving from demonstrations [11], [12], etc. Whereas, since the capabilities and performance of supervised learning deep neural networks are usually largely limited by the available data, the amount of human-labeled data inherently constraints these methods.

With the popularity of high-performance computing, autonomously learning from trial-and-error, namely reinforcement learning, achieves great success in a variety of tasks, especially playing video games like Atari [13]. Deep reinforcement learning can scale to complex 3D manipulation tasks, and learn policies efficiently enough to train on physical robots. Without any prior demonstrations or manually designed representations, a variety of 3D manipulation skills are learned in simulation, even a complex door opening skill on real robots [14]. As the robotic manipulation workspace is fully observed, static, and stable, the applications of deep reinforcement learning are largely limited in robotic manipulation. In the field of mobile robots, the sample space is extremely enlarged by the complex, partially observed environments. Target-driven visual navigation in indoor scenes using deep reinforcement learning simplifies this problem by sampling continuous control actions to discrete actions like forward, left, right [15], which largely limits mobile robot behaviors. Recently, a generalized computation graph is proposed to form a navigation model, which can learn from 64×36 grayscale raw images taken from an onboard forward-facing camera. The car navigates through environments at a fixed speed of 2m/s. The navigation model can offer continuous control actions, namely the steering angle, to direct an RC car in the simulation environment and a real-world environment [16]. This generalized computation graph tries to subsume value-based model-free methods and model-based methods together. Its goal is still to learn collision avoidance policies for mobile robots, which rewards the robot for collision-free navigation.

In recent years, several different methods have been proposed for deep reinforcement learning. Deep Q-learning [17] for the first time combines deep neural networks with reinforcement learning at scale. Deep Q-learning works well on game environments, which is able to master Atari games to superhuman level with only the raw pixels and score as inputs. But, it is poorly understood, and fails on many simple continuous control problems [18], [19]. An asynchronous gradient descent method is proposed to

optimize the deep neural network for continuous motor control problems, which has poor data efficiency and robustness [20]. Trust region policy optimization is not good at dealing with the noise like dropout and parameter sharing, and is also relatively complicated to implement and tune [21]. A new family of policy gradient methods, Proximal Policy Optimization (PPO), are proposed, which have some advantages of trust region policy optimization, but are much easier to implement and show empirically better sample complexity [22], [23]. Simulated robotic locomotion and Atari game playing show that PPO outperforms other online policy gradient methods, and achieves a favorable balance between sample complexity, simplicity, and wall-time [22]. Moreover, it is only evaluated in simulation control agents, and further demonstration and improvement for practical applications on mobile robots are absent, and but required.

In this work, I present a learning-based mapless mobile robot navigation method. A proved efficient policy gradient method, PPO, is utilized for learning continuous control actions. The laser scanning data is first normalized, and then is fed into the proposed method as input. After a few hours of fully autonomous learning, the continuous velocity commands are output to direct a mobile robot running in the simulated environments. The proposed method is implemented as a mobile robot learning system in Python, which organically combines the Robot Operating System (ROS) Framework, OpenAI gym [18], [24], and the Gazebo simulator together. The Gazebo simulator provides a detailed and realistic simulation environment for mobile robots. OpenAI gym offers a toolkit for developing and comparing reinforcement learning algorithms. ROS is a flexible framework for a wide variety of robotic applications. This builds a proper platform for evaluations and demonstrations of the proposed learning-based mobile robot navigation method. The proposed method is trained end-to-end in several different simulation environments to evaluate the performance without any manually designed features, human-provided labels, and prior assumptions. The experiment results show that it can quickly learn policies for collision-free navigation. The output action commands continuously control mobile robots in the environment. Also, the robustness and adaptability of the proposed method are demonstrated in multiple different environments.

In summary, this paper makes the following main contributions:

- A learning-based mapless mobile robot navigation method is presented for learning continuous control tasks with a proved efficient policy gradient method (PPO).
- The presented method is trained end-to-end from scratch in several different simulation environments, in which the performance is evaluated and the robustness and adaptability are demonstrated empirically.
- The proposed learning-based method and mobile robot learning system can be as a general approach to train the mobile robot for more complex continuous tasks in the future.

The remainder of this paper is organized as follows: Section II presents the proposed algorithm in detail. Evaluations in simulation experiments are described in the Section III. Section IV summarizes and concludes the presented work.

II. Proposed Approach

Here, I present an approach to learning continuous control for a mobile robot. The main goal is to learn collision avoidance and stay in the lane policies for mapless mobile robot navigation.

A. Overview

This task is formalized as a reinforcement learning problem, where the mobile robot is rewarded for collision-free navigation, shown in Fig. 2. The state is described by the sparse laser scanning data and the current velocity of the mobile robot. The sparse laser scanning data is directly sampled from the raw laser scanning data with scanning angle 270 degrees. The range data is normalized to $(-1.0, 1.0)$. The mobile robot operates in the environments at a fixed linear speed as in [16]. The current velocity of the mobile robot can be solely described by the current angular velocity. The control actions are learned by trial-and-error during reinforcement learning process, which only include a series of angular velocity

commands. However, at every time step, the angular velocity command is represented by a conditional Gaussian distribution $X \sim \mathcal{N}(\mu, \sigma^2)$ with the mean μ and standard deviation σ. A hyperbolic tangent function *tanh* is employed as the activation function to limit the angular velocity range of the mean μ also in $(-1.0, 1.0)$. The standard deviation σ is constrained by a nonlinear function *softplus* in $(0, +\infty)$. The real output control action is sampled from Gaussian distribution, which achieves a favorable balance between exploitation and exploration. A clip function is applied to the sampled angular velocity, which ensures the sampled angular velocity range in $[-1.0, 1.0]$.

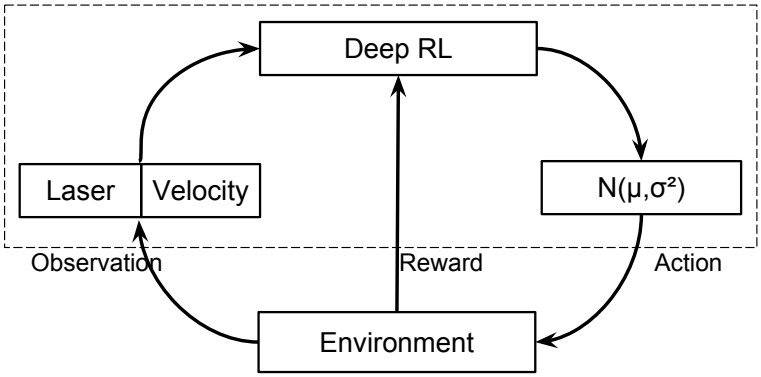

Fig. 2. Overview of learning-based mobile robot navigation method. The state is described by the sparse laser scanning data and the current angular velocity of the mobile robot. The output of deep reinforcement learning network is a conditional Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ to represent the velocity command. At each step, the state, the action, and the reward are collected to fully autonomously train the deep reinforcement learning network.

B. Learning with Proximal Policy Optimization

To learn the current Gaussian distribution of angular velocity for the mobile robot, proximal policy optimization [22], [23] is applied in the work. The PPO algorithm can be considered as an approximate version of trust region policy optimization with first order gradients, which makes it easier for recurrent neural networks (RNNs) in a large-scale setting. Algorithm Box 1 presents the continuous control learning algorithm through PPO in pseudo-code. Actor-critic architecture is used in the proposed algorithm.

Firstly, According to policy π_θ, the mobile robot operates one step in the environment. At each step, the state, the action, and the reward are collected for further training. Then, the advantage function is given by temporal difference (TD) error, which is the difference between discounted rewards $\sum_{t\prime>t} \gamma^{t\prime-t} r_{t\prime}$ and state value $V_\phi(s_t)$. Actor updates θ by a gradient method with respect to $J_{PPO}(\theta)$, which maximizes a surrogate function with the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$. Actor optimizes new policy $\pi_\theta(a_t|s_t)$ based on the advantage function and old policy $\pi_{old}(a_t|s_t)$. The larger the advantage function is, the more likely the new policy changes. However, if the advantage function is too large, the algorithm is very likely to be divergent. Therefore, a KL penalty is introduced to limit the learning rate from old policy $\pi_{old}(a_t|s_t)$ to new policy $\pi_\theta(a_t|s_t)$. Critic updates ϕ by a gradient method with respect to $L_{BL}(\phi)$, which minimizes the loss function of TD error given a data with length-T timesteps. The desired change is set by the hyperparameter KL_{target} in each policy iteration. If the actual change $\text{KL}[\pi_{old}|\pi_\theta]$ belows or exceeds the KL_{target} range in $[\beta_{low}\text{KL}_{target}, \beta_{high}\text{KL}_{target}]$, the scaling term $\alpha > 1$ would adjust the coefficient of $\text{KL}[\pi_{old}|\pi_\theta]$.

Except the KL penalty coefficient to update actor network, another approach which can be used as an alternative is the clipped surrogate objective [22]. The main objective can be described as

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{1}$$

Algorithm 1: Learning Continuous Control through Proximal Policy Optimization

for *i = 1,...,N* **do**
  Run policy π_θ for T timesteps, collecting s_t, a_t, r_t
  Estimate advantages $\hat{A}_t = \sum_{t\prime>t} \gamma^{t\prime-t} r_{t\prime} - V_\phi(s_t)$
  $\pi_{old} \leftarrow \pi_\theta$
  for *j = 1,...,M* **do**
    $J_{PPO}(\theta) = \sum_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} \hat{A}_t - \lambda \mathrm{KL}[\pi_{old}|\pi_\theta]$ Update θ by a gradient method w.r.t. $J_{PPO}(\theta)$
  end
  for *j = 1,...,B* **do**
    $L_{BL}(\phi) = -\sum_{t=1}^{T}(\sum_{t\prime>t} \gamma^{t\prime-t} r_{t\prime} - V_\phi(s_t))$ Update ϕ by a gradient method w.r.t. $L_{BL}(\phi)$
  end
  if $KL[\pi_{old}|\pi_\theta] > \beta_{high} KL_{target}$ **then**
    $\lambda \leftarrow \alpha\lambda$
  else if $KL[\pi_{old}|\pi_\theta] < \beta_{low} KL_{target}$ **then**
    $\lambda \leftarrow \lambda/\alpha$
  end
end

where $\epsilon = 0.2$ is hyperparameter. The clip term $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$ has the same motivation as the KL penalty, which is also used to limit too large policy updates.

C. Reward Function

To simplify the reward function, the critic network uses only two different conditions without normalization or clipping, which can be defined as

$$r_t(s_t, a_t) = \begin{cases} r_{move} & \text{if not collision} \\ r_{collision} & \text{if collision} \end{cases} \tag{2}$$

If the mobile robot freely operates in the environment, a positive reward r_{move} is provided. Otherwise, if the mobile robot collides with the obstacle through a minimum laser scanning range checking, a large negative reward $r_{collision}$ is given. This reward function motivates the mobile robot to navigate through the environment without collision, and stay in the lane.

III. Simulation Based Evaluation

In order to systematically evaluate the performance of the proposed learning-based mobile robot mapless navigation method, simulation experiments are performed. The proposed method can freely navigate the mobile robot through multiple simulation environments only using the sparse laser scanning data. The mobile robot can quickly learn to navigate complex environments after a few hours autonomously training. Videos of the experiments can be found at https://youtu.be/P0bwzXI4EEA.

A. Simulation Setup

Since the proposed learning-based mobile robot mapless navigation method operates in a closed loop with states observing by the robot's perception, rewards from the environment, policy learning with Deep RL, and actions applied to motion control (see Fig. 2), a detailed and realistic simulation is required.

The Gazebo simulator 7.0.0 is adopted in the training process for more realistic virtual environment simulation. A kobuki-based turtlebot is used as the mobile robot platform. The mobile robot receives the laser scanning data from a ULM-30LX Hokuyo Laser Range Finder, which has a field of view (FOV)

with 270 degrees and angular resolution of 1.0 degree. The scanning distance ranges from 0.1m to 10m. A toolkit for reinforcement learning using ROS and the Gazebo simulator extends the OpenAI gym for mobile robot learning. Although the deep reinforcement learning process and Gazebo simulation require considerable computational resources, there is no special GPU unit used in this simulation. To build practical and low computational cost algorithm, an ordinary GPU, namely GeForce GT 610, is employed in our evaluation. The proposed method is tested in the virtual simulation environment on an ordinary PC with 3.4 GHz six-core Intel i7 processor and 64 GB memory. The mobile robot learning system is implemented in the Python language and is run in the Robot Operating System (ROS) Kinetic on Ubuntu 16.04 LTS (Xenial).

The mobile robot training related parameters are presented as follows. Considering the physical dynamic of the turtlebot, the fixed linear velocity is set to the maximum linear velocity of $0.65m/s$, and the angular velocity ranges in $[-1.0, 1.0]rad/s$. Although the angular velocity is sampled from a Gaussian distribution, it is also clipped to $[-1.0, 1.0]rad/s$. The distance threshold of the laser range, which determines whether crashing to the wall, is set to $0.2m$. The maximum iteration steps for each episode is set to 3000. The actor and critic neural network are both implemented by 3 fully-connected neural network layers with 512 nodes. And also, the same learning rate $\alpha = 0.00002$ are used both for actor and critic neural network. Collecting the state, the reward, and the action for every 16 times, the batch is executed to update learning policy by a gradient method. For each end of the episode, the learning policy is also updated.

Three different simulation scenarios are used to evaluate the proposed method. Since the main goal is to learn collision avoidance and stay in the lane policies for mapless mobile robot navigation, the closed loop environments are chosen for the mobile robot training. The simulation scenarios include the square-shaped environment, the round-shaped environment, and the L-shaped environment, as shown in Fig. 1.

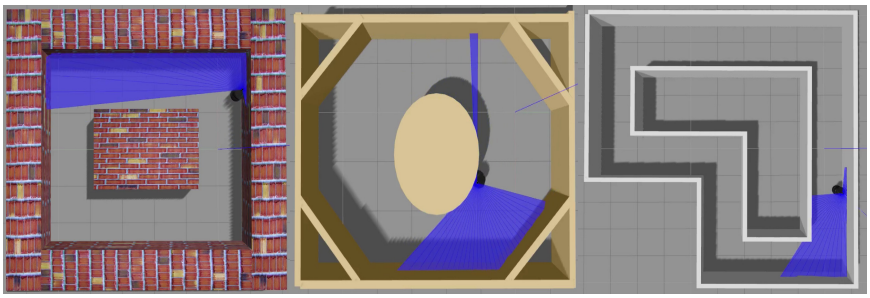

Fig. 3. Example failure cases during training process in different virtual environment simulated by Gazebo.

B. Simulation Results

I present results evaluating the proposed learning-based mobile robot navigation method on multiple different simulated mobile robot scenarios. The mobile robot performs motion planning according to the laser scanning data input. After iterative trial-and-error learning, proper actions are sent to control the mobile robot for collision-free navigation. However, sometimes the policy fails shown in Fig. 3. The successful policy drives the mobile robot follow the path through the L-shaped environment without collision shown in Fig. 6. More detailed experimental information can be seen from the video.

1) The Square-Shaped Environment: The square-shaped environment is presented on the left of Fig 1. The task is to run in a square lane without collision. Fig. 4 shows the results obtained through 300 episodes. Fig. 4A shows the cumulated rewards over episodes. According to the blue line, after 200 episodes, the mobile robot already can make a maximum of 3000 iterations. Since during training process the action is sampled from a Gaussian distribution to achieve a reinforcement learning balance between exploitation

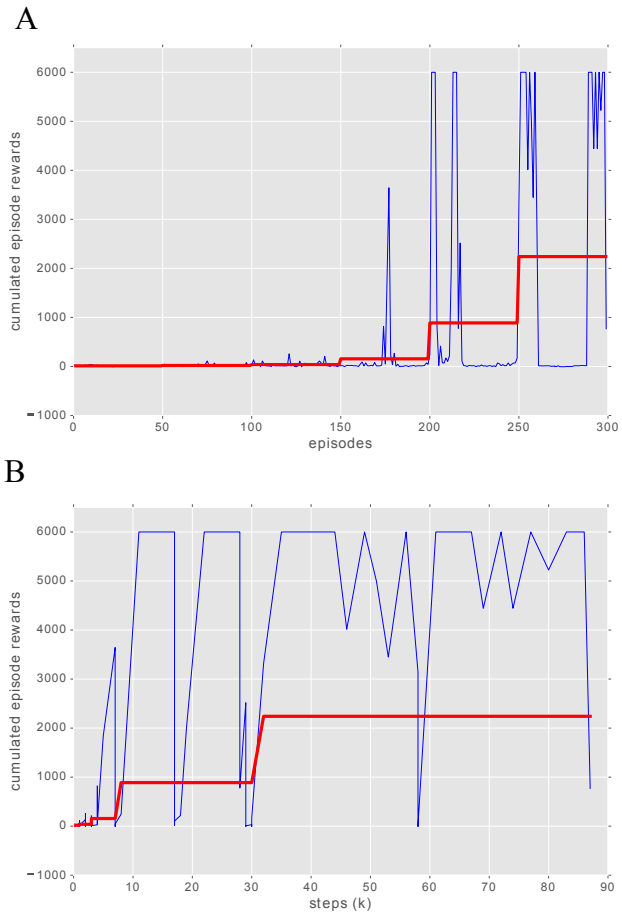

Fig. 4. The cumulated reward graph over episodes (A) and iteration steps (B) obtained from the monitoring of the square-shaped environment. The blue line shows the cumulated rewards while the red line shows the averaged rewards.

and exploration, sometimes the exploration would make the mobile robot cumulate fewer rewards shown in Fig. 4. The same situations would also be shown in other simulated environments.

However, after a period of learning, the latter episode runs more iteration steps. To clearly show the reward cumulating process, Fig. 4B shows the cumulated reward graph over iteration steps. After $10k$ iteration steps, the mobile robot can reach the maximum of 3000 iterations without collision. After $30k$ iteration steps, the average cumulated rewards exceed 2000. From Fig. 4B, only few iteration steps corresponds to very low cumulated rewards, and most of them are more than 4000. Cumulated rewards around 4000 or higher usually mean that the robot did not crash more than 20 laps for the square-shaped environment.

2) The Round-Shaped Environment: The middle of Fig. 1 is a bird's-eye view of the round-shaped environment. Fig. 5 shows that after only 90 episodes, the mobile robot already can run the maximum of 3000 iterations without collision. After $20k$ iteration steps, the average cumulated reward is nearly 3000.

3) The L-Shaped Environment: The L-Shaped Environment is a typical indoor environment, like a corridor in a building, shown in the right of Fig. 1. Fig. 6 shows the screenshots when the mobile robot runs following the path through the L-shaped environment in a loop. Fig. 7 shows the results obtained from the monitoring of the L-shaped environment. After 200 episodes, the cumulated reward is nearly closed to 1000. Although after $20k$ iteration steps, one of the episodes has reached the maximum of 3000 iterations, the rewards are slightly unstable. The possible reason is that the L-shaped environment is much bigger and more complex than the square-shaped environment and the round-shaped environment, the

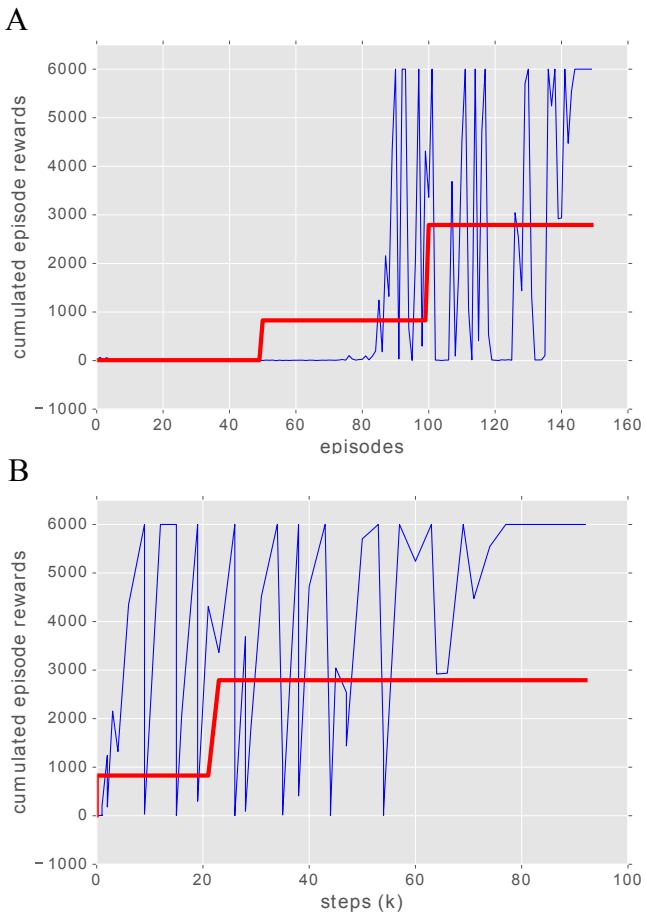

Fig. 5. The cumulated reward graph over episodes (A) and iteration steps (B) obtained from the monitoring of the round-shaped environment. The cumulated rewards depict by the blue line, and the averaged rewards are shown by the red line.

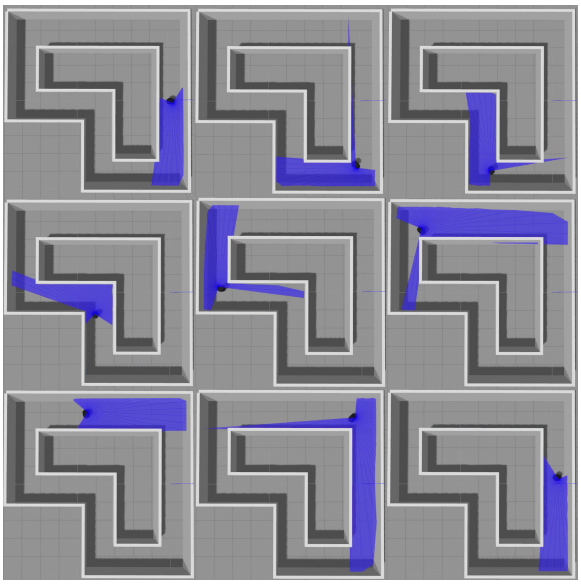

Fig. 6. The mobile robot runs following the path through the L-shaped environment in a loop.

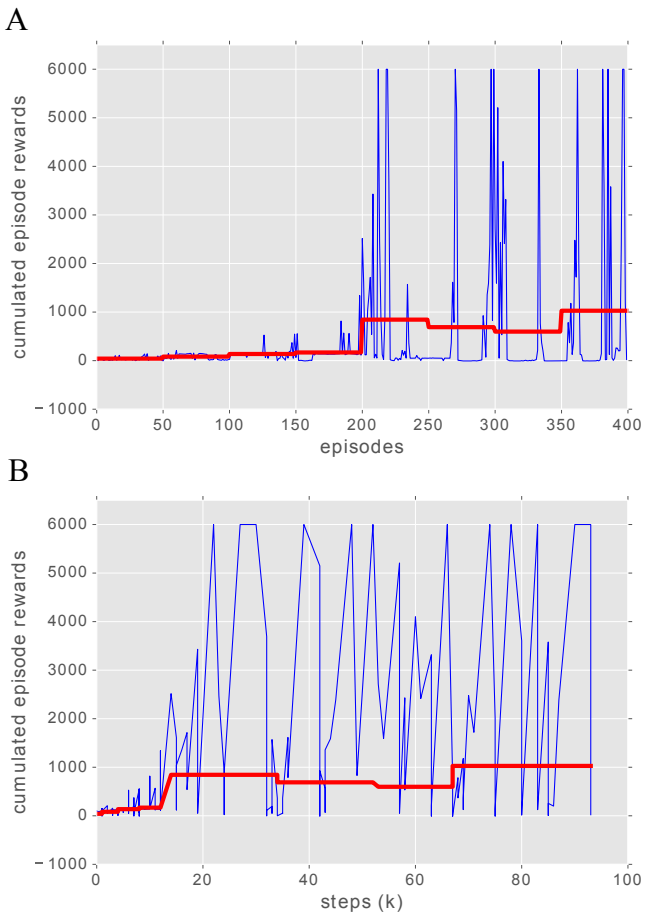

Fig. 7. The cumulated reward graph over episodes (A) and iteration steps (B) obtained from the monitoring of the L-shaped environment. The blue line and the red line represent the cumulated rewards and the averaged rewards, respectively.

Gaussian distribution over actions would be easier to make the mobile robot collide with the wall. The successful evaluation process can be seen from videos of the experiments.

IV. Summary & Conclusion

In this work, a learning-based mapless mobile robot navigation method is presented that is capable of learning continuous control through proximal policy optimization from continuous state spaces. By taking the normalized laser scanning data as input, after trial-and-error learning, it can directly generate the continuous velocity commands to control a mobile robot for collision-free navigation. The ROS framework, OpenAI gym, and the Gazebo simulator are organically combined to provide a platform for the implementation of the proposed method. In three different environments, including the square-shaped environment, the round-shaped environment, and the L-shaped environment, the proposed method is trained end-to-end to evaluate the performance without any manually designed features, human-provided labels, and prior assumptions. The simulation experiments show that after a few hours of fully autonomous training, the continuous velocity commands are successfully learned for mobile robots. The proposed method can be further extended for more complex continuous tasks.

Although the proposed method is tested to learn to avoid collisions in multiple different environments, the control actions are simplified at a fixed linear speed to move the mobile robot through the simulated environment. The future work would extend the proposed method to learn angular velocity and linear velocity at the same time, and deploy it to the real mobile robot in the complex physical environment. Also,

further research would investigate how to learn in large outdoor environments with dynamic obstacles. The mobile robot learning method may further provide an efficient and robust way for practical mobile robot applications.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China (No. 2016YFC0801808).

REFERENCES

[1] U. Nehmzow, *Mobile robotics: a practical introduction*. Springer Science & Business Media, 2012.
[2] R. C. Arkin, "Motor schemabased mobile robot navigation," *The International journal of robotics research*, vol. 8, no. 4, pp. 92–112, 1989.
[3] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 2, pp. 237–267, 2002.
[4] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
[5] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 163–168.
[6] 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.
[7] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*. Ieee, 1991, pp. 1442–1447.
[8] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.
[9] D. Barnes, W. Maddern, and I. Posner, "Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 203–210.
[10] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.
[11] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.
[12] 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.
[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[14] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3389–3396.
[15] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3357–3364.
[16] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.
[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
[19] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.
[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.
[21] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.
[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[23] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.
[24] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo," *arXiv preprint arXiv:1608.05742*, 2016.