

## Research Article

# Visual Navigation with Asynchronous Proximal Policy Optimization in Artificial Agents

Fanyu Zeng  and Chen Wang

School of Computer Science and Engineering, Center for Robotics, University of Electronic Science and Technology of China, Chengdu 611731, China

Correspondence should be addressed to Fanyu Zeng; zengfanyu\_cs@163.com

Received 12 February 2020; Revised 10 August 2020; Accepted 21 September 2020; Published 15 October 2020

Academic Editor: Weitian Wang

Copyright © 2020 Fanyu Zeng and Chen Wang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Vanilla policy gradient methods suffer from high variance, leading to unstable policies during training, where the policy's performance fluctuates drastically between iterations. To address this issue, we analyze the policy optimization process of the navigation method based on deep reinforcement learning (DRL) that uses asynchronous gradient descent for optimization. A variant navigation (asynchronous proximal policy optimization navigation, *appoNav*) is presented that can guarantee the policy monotonic improvement during the process of policy optimization. Our experiments are tested in DeepMind Lab, and the experimental results show that the artificial agents with *appoNav* perform better than the compared algorithm.

## 1. Introduction

Navigation in an unstructured environment is one of the most important abilities for mobile robotics and artificial agents [1–3]. Traditional methods mainly divide navigation into several parts [4]: simultaneous localization and mapping (SLAM) [5–7], path planning [8], and semantic segmentation [9, 10]. The methods mentioned are not an end-to-end algorithm where each part is a challenging research subject, and the fusion of each part often leads to large computational errors. To reduce the fusion error, we focus on the end-to-end navigation based on deep reinforcement learning where navigational abilities could emerge as the byproduct of an artificial agent learning policy with reward maximization.

With the fast development of deep learning [11–14], a variety of DRL architectures have been proposed [2]. Mnih et al. [15] presented the advances in training deep neural networks to develop the deep Q-network (DQN), which can learn successful policies directly from high-dimensional image inputs using end-to-end reinforcement learning. On-policy reinforcement learning methods such as actor-critic (AC) [16, 17] were proposed such that the actor is the policy, and the critic is the baseline. Minh et al. [18] presented

asynchronous variants of AC algorithms, termed as asynchronous advantage actor-critic (A3C), and showed that parallel actor-learners have a stabilizing effect on training artificial agents. Researchers can construct navigation agents based on these DRL algorithms. However, vanilla policy gradient methods have poor data efficiency [19], which leads to navigation agents suffering from high variance and unstable policies.

In this work, we take A3C as an example to show how to guarantee the policy monotonic improvement. The training environment is DeepMind Lab [20], and it is a first-person 3D virtual environment designed for research and development of general artificial intelligence. DeepMind Lab can be used to study how autonomous artificial agents learn complex tasks in large, partially observed, and visually diverse worlds. In addition, the worlds are rendered with rich science fiction-style visuals. Actions are to look around and move in the 3D virtual world, and example tasks include navigation in different mazes. Mirowski et al. [21] proposed a DRL navigation method based on A3C [18], augmented with auxiliary learning targets, to train artificial agents to navigate in DeepMind Lab. For ease of expression, we call the DRL navigation using A3C as *a3cNav*.

In this paper, the issues on policy optimization for navigation based on the vanilla policy gradient are analyzed; this type of navigation cannot control the change of expected advantage when an artificial agent learns to navigate in a maze. Based on the navigation techniques presented in [21], we show how to reduce training variances and get higher reward when an artificial agent interacts with an environment. Inspired by [19, 22], we adjust the policy update process of the navigation in [21] to guarantee the monotonic improvement of the navigation policy. Experimental results show that an artificial agent via *appoNav* learns better navigation policy in DeepMind Lab and suffers from lower standard deviation than *a3cNav*.

## 2. Related Work

Traditional navigation, which is model-based, includes simultaneous localization and mapping (SLAM) [5, 7, 23], path planning [8, 24], and semantic segmentation [9]. Each part of them is a challenge research area, and the fusion of them often leads to large computation error. Moreover, model-based navigation needs to model the environments effectively for some dynamic and complex scenes, which severely affect navigation performance.

With recent advances in DRL, many navigation methods based on DRL have been proposed [2]. DRL navigation, which is end to end, avoids the computation error caused by the fusion of traditional navigation. Mirowski et al. [21] addressed navigation via auxiliary depth prediction and loop-closure classification tasks. Jaderberg et al. [25] also used auxiliary tasks for navigation and incorporated A3C with control tasks and prediction tasks including pixel control and reward prediction. By using features extracted from the world model as inputs to an agent, Ha and Schmidhuber [26] used DRL to construct a world model and used the model in a car navigation task. Bruce et al. [27] leveraged an interactive world model based on DRL built from a single traversal of the environment and utilized a pretrained visual feature encoder to demonstrate successful zero-shot transfer under real-world environmental variations without fine-tuning. Banino et al. [28] proposed a vector-based navigation method that fuses DRL with grid-like representations in the artificial agent. When these DRL navigation agents interact with environments, the state sequences of each interaction change a lot, leading to large fluctuations in rewards. Therefore, these DRL navigation methods suffer from high variance and have unstable policies during training.

## 3. Background

**3.1. Reinforcement Learning.** We consider the standard reinforcement learning setting where an artificial agent interacts with an environment over a number of discrete time steps. At each time step  $t$ , the agent receives a state  $s_t$  from the environment and outputs an action  $a_t$  according to its learned policy  $\pi$ . In return, the environment gives the agent a next state  $s_{t+1}$  and a reward  $r_t$ . The goal of reinforcement learning is to maximize the accumulated reward

$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ , which is a discounted sum of rewards. The action-value function  $Q^\pi = \mathcal{E}[R_t | s_t = s, a]$  is the expected return following action  $a$  from state  $s$  under policy  $\pi$ . The value function  $V^\pi = \mathcal{E}[R_t | s_t = s]$  is the expected return from state  $s$ .

In policy-based methods, let  $\pi(a | s; \theta)$  be a policy with parameters  $\theta$ , which is updated by performing gradient ascent on  $\mathcal{E}[R_t]$ . Policy gradient algorithms adjust the policy by updating parameters  $\theta$  in the direction  $\nabla_{\theta} \log \pi(a_t | s_t; \theta) R_t$  that is an unbiased estimate of  $\nabla_{\theta} \mathcal{E}[R_t]$ . To reduce the variance of this estimate, Williams [29] subtracted a learned function called baseline  $b_t(s_t)$  for the return, so the improved gradient becomes  $\nabla_{\theta} \log \pi(a_t | s_t; \theta) (R_t - b_t(s_t))$ . There exists an equation  $b_t(s_t) \approx V^\pi(s)$ , and  $R_t - b_t(s_t)$  can be seen as an estimate of the advantage of action at under state  $s_t$ . The numerical value of  $Q^\pi(s, a)$  equals the value of  $R_t$ ; hence, the advantage function can be rewritten as  $A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$ . This method is called actor-critic (AC) architecture where the actor is the policy  $\pi$  and the critic is the baseline  $b_t$  [16, 17]. Minh et al. [18] presented asynchronous variants of AC algorithms, termed as asynchronous advantage actor-critic (A3C), and showed that parallel actor-learners have a stabilizing effect on training artificial agents.

When a DRL agent interacts with its environment, the state sequences of each interaction change a lot, leading to fluctuations in rewards. Therefore, DRL algorithms (such as DQN and A3C) have unstable fluctuations during training. Researchers wonder whether they can find a method to reduce such fluctuations while maintaining a steady improvement in the policy. Schulman et al. [22] proposed trust region policy optimization (TRPO) to make the monotonic improvement for the policy. Furthermore, Schulman et al. [19] proposed proximal policy optimization (PPO) to simplify the calculation of TRPO. In addition, Heess et al. [30] proposed a distributed implementation of PPO, called distributed PPO. Besides the similar process of the gradient update with A3C, distributed PPO includes various tricks, such as normalizations (observation normalization, reward reshape normalization, and per-batch normalization of the advantages), sharing of algorithm parameters across local workers, and additional trust region constraint. These tricks result in that the computation of distributed PPO is more complex than *appoNav*.

**3.2. NavA3C +  $D_1D_2$ .** In this work, we use the *NavA3C +  $D_1D_2$*  architecture [21] as shown in Figure 1, which includes 2 CNNs and 2 LSTMs. *NavA3C +  $D_1D_2$*  has 4 inputs: the current RGB image  $x_t$ , previous reward  $r_{t-1}$ , previous action  $a_{t-1}$ , and the current velocity  $v_t$ . The 2 CNNs act as the encoder for RGB image  $x_t$ , and the first LSTM makes associations between reward  $r_{t-1}$  and visual observations  $x_t$  that are provided as context to the second LSTM from which the policy  $\pi(a_t | s_t; \theta)$  and the value  $V(s_t; \theta_v)$  are computed. Artificial agents based on this architecture try to maximize the cumulative reward  $R_t$  during their interaction with the maze and minimize the auxiliary depth losses  $L_{\text{Depth1}}$  and  $L_{\text{Depth2}}$ . Finally, the agent can learn how to

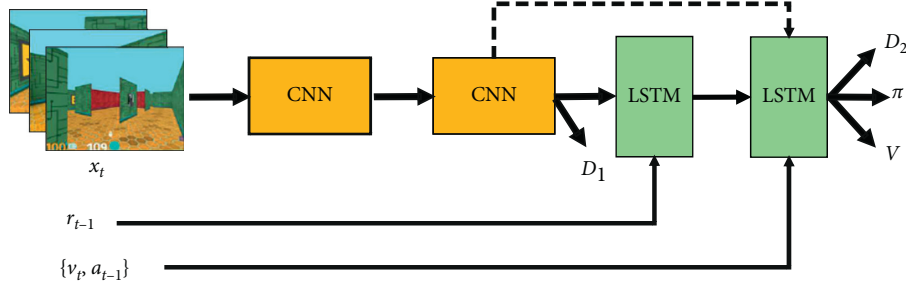


FIGURE 1: *a3cNav* architecture. In the architecture, image  $x_t$  is the input of *a3cNav*, and following the full connection layer is a two-layer CNN which outputs depth  $D_1$  as well as a two-layer stacked LSTM which outputs depth  $D_2$ , policy  $\pi$ , and value  $V$ . In addition, auxiliary task used in this architecture in which the first LSTM only receives the reward and the velocity and previously selected action are fed into the second LSTM.

navigate in DeepMind Lab. For ease of expression, we rename *NavA3C* +  $D_1D_2$  as *a3cNav*.

*a3cNav* is based on the A3C framework into which unsupervised auxiliary tasks are incorporated. Therefore, its loss function includes the loss of A3C  $L_{A3C}$  and the loss of auxiliary tasks. *a3cNav* can be optimized as follows:

$$L_{a3cNav}(\theta) = L_{A3C} + \lambda_{Depth1}L_{Depth1} + \lambda_{Depth2}L_{Depth2}, \quad (1)$$

where  $\lambda_{Depth1}$  and  $\lambda_{Depth2}$  are weighting terms on the individual loss components.

The global parameters  $\theta$  of *a3cNav* are updated in multithread environments, and  $\theta$  are copied to the local worker parameters  $\theta'$ . The local worker of *a3cNav* interacts with the maze, and the policy gradients wrt  $\theta'$  and the value gradients wrt  $\theta'_v$  are computed from the policy loss and value loss. The gradient for the parameter update is proportional to the product of advantage function  $A_t$ . Equation (2) shows the calculation of gradients:

$$\begin{aligned} d\theta &\leftarrow d\theta + \nabla_{\theta'} \log \pi(a_t | s_t; \theta') (R - V(s_t; \theta'_v)) \\ &\quad + \beta \nabla_{\theta'} H(\pi(s_t; \theta')) \\ d\theta_v &\leftarrow d\theta_v + \frac{\partial (R - V(s_t; \theta'_v))^2}{\partial \theta'_v} \end{aligned}, \quad (2)$$

where  $H(\pi(s_t; \theta'))$  is the entropy of the policy  $\pi$ , which improves exploration by discouraging premature convergence to suboptimal deterministic policies. Then, asynchronous update of  $\theta$  using  $d\theta$  and of  $\theta_v$  using  $d\theta_v$  are applied into the global network for parameter update.

## 4. Approach

**4.1. Monotonic Policy Improvement.** The artificial agent interacts randomly with the environment which in turn gives high-dimensional images to the agent. Hence, *a3cNav* has poor data efficiency and robustness. In addition, complex navigation environment that sends changing images to the artificial agent aggravates the variance and instability of training. In detail, each local worker of *a3cNav* interacts with the maze, and the gradients with big variance are applied to

the global network of *a3cNav*, leading to the unstable training of the agent. In this section, we improve the parameter updates of *a3cNav* to guarantee its policy monotonic improvement.

In [22], a policy can be rewritten as

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a | s) A_{\pi}(s, a), \quad (3)$$

where  $\pi$  denotes a stochastic policy and  $\tilde{\pi}$  is another policy.  $\eta(\pi)$  and  $\eta(\tilde{\pi})$  are the expected discounted cost for  $\pi$  and  $\tilde{\pi}$ , respectively. Here,  $\rho_{\tilde{\pi}}(s)$  is the distribution of the state  $s$  according to  $\tilde{\pi}$ , and  $A_{\pi}$  is the advantage function following  $\pi$ .

Equation (3) implies that if we want to reduce  $\eta$  or leave it as constant, we should keep the expected advantage  $\sum_a \tilde{\pi}(a | s) A_{\pi}(s, a) \leq 0$  at every state  $s$  when a policy update  $\tilde{\pi} \rightarrow \pi$ . This demonstrates that if we want to reduce the training variance of *a3cNav* and keep its policy monotonic improvement, we must guarantee  $\sum_a \tilde{\pi}(a | s) A_{\pi}(s, a) \leq 0$ . However, *a3cNav* cannot control the change of the expected advantage when the artificial agent learns to navigate in the maze.

To make the policy monotonic improvement, Schulman et al. [22] proposed a trust region constraint, as shown in equation (4), over policy update to make  $\sum_a \tilde{\pi}(a | s) A_{\pi}(s, a) \leq 0$ :

$$\max_{\theta} \mathbb{E}_t \left[ \frac{\pi_{\theta}(a_t | s_t) \hat{A}_t}{\pi_{\theta_{old}}(a_t | s_t)} \right], \quad (4)$$





























































$$\mathbb{E}_t \left[ \text{KL}[\pi_{\theta_{old}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)] \right] \leq \delta.$$

Equation (4) is relatively complex and is not compatible with the architectures which include parameter sharing between the policy function and the value function, or with auxiliary tasks [19]. The policy and the value network of *a3cNav* both share the same network, and *a3cNav* has the auxiliary depth prediction. Therefore, TRPO cannot be used into *a3cNav*.

$$\mathcal{E}_t \left[ \min \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right]. \quad (5)$$

PPO [19] improves TRPO with only first-order optimization and replaces the constraint with the clipped

TABLE 1: The states that the artificial agent sees in *stairway\_to\_melon*.

Time Episode	600	700	800	900	1000	1100	1200	1300	1400	1500
The first episode										
The second episode										
The third episode										
Time Episode	1600	1700	1800	1900	2000	2100	2200	2300	2400	2500
The first episode										
The second episode										
The third episode										

surrogate objective as equation (5). Hence, PPO is a first-order optimization method and is compatible with parameter sharing and auxiliary tasks.

**4.2. *appoNav*.** To make the monotonic improvement for the navigation policy, we seek to incorporate the features of PPO into the local worker of *a3cNav*. In each thread, the improved local policy tends to improve monotonically. And the new local gradients are applied to the global network, leading to the whole network with monotonic improvement. As the navigation method is based on the monotonic policy improvement of PPO, we call this navigation as *appoNav*.

Assume that the global network shared parameter vector  $\theta$  and local worker parameter vector  $\theta'$ . Equation (6) is the policy optimization loss of A3C [18]:

$$L_{A3C} = \log \pi(a_t | s_t; \theta) + \beta H(\pi(s_t; \theta)). \quad (6)$$

When added to the local worker of *a3cNav*, the loss function becomes the form of equation (5) with entropy of the policy, and it is rewritten for the local workers as

$$\mathbb{E}_t \left[ \min \left( \frac{\pi_{\theta'}(a_t | s_t)}{\pi_{\theta'_{old}}(a_t | s_t)} \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta'}(a_t | s_t)}{\pi_{\theta'_{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] + \beta H(\pi(s_t; \theta')). \quad (7)$$

Equation (7) is the policy update of the local worker of *a3cNav*, that is, *appoNav*. Each local worker has a low variance than before and applies the new gradient to the global network for the policy update. Finally, the whole

policy generated by *appoNav* has lower variance and more stable training performance.

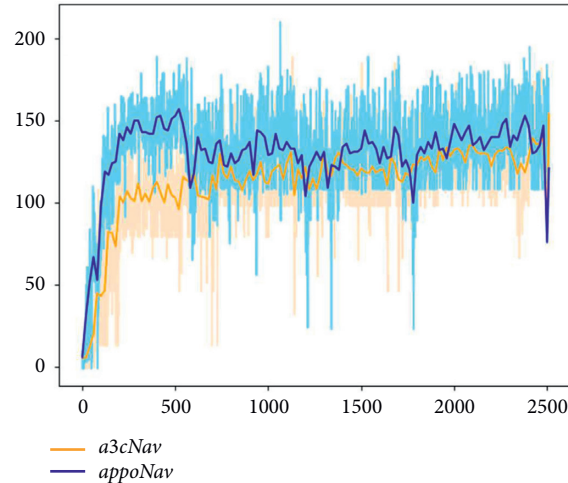
## 5. Experiments

**5.1. Experimental Settings.** We implement our algorithm in TensorFlow and train it on Nvidia GeForce GTX Titan X GPU and Intel Xeon E5-2687W v2@3.4GHz\*17 CPU.

The proposed method is evaluated in DeepMind Lab environments [20]. The action space in DeepMind Lab has 8 actions: the agent can rotate in small increments, accelerate forward or backward or sideways, or induce rotational acceleration while moving. Reward encourages the agent to learn navigation; a reward is achieved when the artificial agent reaches a goal from a random start location and orientation. If the agent reaches the goal, a new episode starts, and the same interaction restarts. Fruit represents the reward in DeepMind Lab: apples are worth 1 point, strawberries 2 points, and goals 10 points.

*appoNav* is evaluated by training the agent in *stairway\_to\_melon* and *nav\_maze\_static\_01* of DeepMind Lab. For ease of expression, we name *stairway\_to\_melon* as the stairway maze and *nav\_maze\_static\_01* as the static01 maze. In each case, blue curve stands for *a3cNav* and orange for *appoNav*. For experimental analysis, we run 2500 episodes for the stairway maze and 7800 episodes for the maze01 maze.

**5.2. Experimental Results and Analysis.** Table 1 shows the images that the artificial agent sees in the stairway maze; we stochastically select 3 episodes from time 600 to 2500 with

FIGURE 2: Reward achieved by the artificial agent in *stairway\_to\_melon*.TABLE 2: Standard deviation of the reward in *stairway\_to\_melon*.

Algorithm	Standard deviation
<i>a3cNav</i>	30.16
<i>appoNav</i>	27.24

TABLE 3: The states that the artificial agent sees in *stairway\_to\_melon*.

Time Episode	1000	1200	1400	1600	1800	2000	2200	2400	2600	2800
The first episode										
The second episode										
The third episode										
Time Episode	3000	3200	3400	3600	3800	4000	4200	4400	4600	4800
The first episode										
The second episode										
The third episode										

interval 100, which demonstrate three different states at the same time with different episodes. The artificial agents can receive different images and be not stuck in one place, which demonstrates the agents learning to navigation in stairway maze.

Figure 2 shows the reward achieved by the artificial agent in *stairway\_to\_melon*; it shows that *appoNav* gets higher reward than *a3cNav*. In addition, we calculate the standard deviation (std) of the reward curve. From Table 2, the reward

std of *appoNav* and *a3cNav* is 27.24 and 30.16, respectively; this shows that the learning process of the former is more stable than the latter one.

The reason why our method converges faster is that the local worker of *appoNav* can generate a more stable policy with the monotonic improvement when it interacts with the stairway. During the training iterations, improved accumulated gradients are applied for the parameter update of *appoNav*, which make *appoNav* more stable than *a3cNav*.

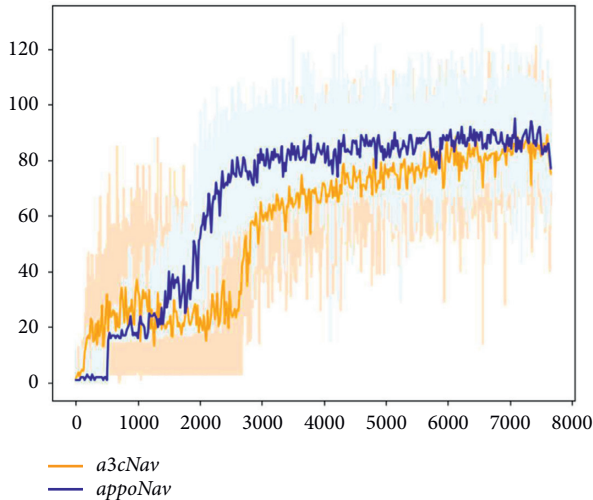


FIGURE 3: Reward achieved by the artificial agent in *nav\_maze\_static\_01*.

TABLE 4: Standard deviation of the reward in *nav\_maze\_static\_01*.

Algorithm	Standard deviation
<i>a3cNav</i>	28.99
<i>appoNav</i>	24.79

For further verification of *appoNav*'s effectiveness, we test our agent in the maze01 maze which is more complex than the stairway maze. Because the agent needs more time to converge, we stochastically select 3 episodes from time 1000 to 4800 with interval 200, as shown in Table 3.

Figure 3 shows the reward achieved by the artificial agent in *nav\_maze\_static\_01*; it demonstrates that *appoNav* performs better than *a3cNav*, and it has higher reward. Table 4 shows that the std of *a3cNav* is 28.99, and the std of *appoA3C* is 24.79. The policy learnt by *appoNav* is more stable than the policy learnt by *a3cNav*.

Owing to that *appoNav* uses better gradient ascents to update each policy, the artificial agent with *appoNav* learns stronger navigation ability as each local worker produces a more stable policy in the complex maze.

## 6. Conclusion

Visual navigation-based vanilla policy gradient methods suffer from high variance and instability during training, where the navigation performance fluctuates greatly between iterations. We analyze the reason why visual navigation suffers such an issue and improve its policy update to guarantee the policy monotonic improvement. The improved method *appoNav* has lower standard deviation and gets higher reward. In short, *appoNav* can learn better navigation policy.

## Data Availability

The raw data required to reproduce these findings are available to download from <https://github.com/deepmind/lab>.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (U1813202, 61773093, and 62003381), National Key R&D Program of China (2018YFC0831800), Research Programs of Sichuan Science and Technology Department (17ZDYF3184), and Important Science and Technology Innovation Projects in Chengdu (2018-YF08-00039-GX).

## References

- [1] 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.
- [2] F. Zeng, C. Wang, and S. S. Ge, "A survey on visual navigation for artificial agents with deep reinforcement learning," *IEEE Access*, vol. 8, Article ID 135426, 135442 pages, 2020.
- [3] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, Cambridge, MA, USA, 2005.
- [4] C. Cadena, L. Carlone, H. Carrillo et al., "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.
- [5] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: large-scale direct monocular slam," in *Proceedings of the European Conference on Computer Vision*, pp. 834–849, Springer, Zurich, Switzerland, September 2014.
- [6] H. Lategahn, A. Geiger, and B. Kitt, "Visual slam for autonomous ground vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1732–1737, IEEE, Shanghai, China, June 2011.
- [7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [8] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.
- [9] Y. Zhang, H. Chen, Y. He, M. Ye, X. Cai, and D. Zhang, "Road segmentation for all-day outdoor robot navigation," *Neurocomputing*, vol. 314, pp. 316–325, 2018.
- [10] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3234–3243, Las Vegas, NV, USA, June 2016.
- [11] X. Li, M. Ye, Y. Liu, and C. Zhu, "Adaptive deep convolutional neural networks for scene-specific object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 9, pp. 2538–2551, 2017.
- [12] X. Li, M. Ye, Y. Liu, F. Zhang, D. Liu, and S. Tang, "Accurate object detection using memory-based models in surveillance scenes," *Pattern Recognition*, vol. 67, pp. 73–84, 2017.
- [13] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Transfer independently together: a generalized framework for domain adaptation," *IEEE Transactions on Cybernetics*, vol. 49, no. 6, pp. 2144–2155, 2018.

- [14] J. Li, Y. Wu, and K. Lu, "Structured domain adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1700–1713, 2016.
- [15] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
- [16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA, 2018.
- [17] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proceedings of the 2012 American Control Conference (ACC)*, pp. 2177–2182, IEEE, Montreal, Canada, June 2012.
- [18] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937, New York, NY, USA, June 2016.
- [19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, <https://arxiv.org/abs/1707.06347>.
- [20] C. Beattie, J. Z. Leibo, D. Teplyaev et al., "Deepmind lab," <https://arxiv.org/abs/1612.03801>.
- [21] P. Mirowski, R. Pascanu, F. Viola et al., "Learning to navigate in complex environments," in *Proceedings of the International conference on Learning Representations*, San Juan, PA, USA, May 2016.
- [22] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the International Conference on Machine Learning*, pp. 1889–1897, Lille, France, July 2015.
- [23] S. S. Ge, Q. Zhang, A. T. Abraham, and B. Rebsamen, "Simultaneous path planning and topological mapping (sp2atm) for environment exploration and goal oriented navigation," *Robotics and Autonomous Systems*, vol. 59, no. 3-4, pp. 228–242, 2011.
- [24] S. S. Ge, X. Lai, and A. A. Mamun, "Boundary following and globally convergent path planning using instant goals," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 2, pp. 240–254, 2005.
- [25] M. Jaderberg, V. Mnih, W. M. Czarnecki et al., "Reinforcement learning with unsupervised auxiliary tasks," 2016, <https://arxiv.org/abs/1611.05397>.
- [26] D. Ha and J. Schmidhuber, "World models," 2018, <https://arxiv.org/abs/1803.10122>.
- [27] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "One-shot reinforcement learning for robot navigation with interactive replay," in *Proceedings of the Conference and Workshop on Neural Information Processing Systems*, Long Beach, CA, USA, December 2017.
- [28] A. Banino, C. Barry, B. Uria et al., "Vector-based navigation using grid-like representations in artificial agents," *Nature*, vol. 557, no. 7705, pp. 429–433, 2018.
- [29] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.
- [30] N. Heess, D. TB, S. Sriram et al., "Emergence of locomotion behaviours in rich environments," 2017, <https://arxiv.org/abs/1707.02286>.