A Brief Survey: Deep Reinforcement Learning in Mobile Robot Navigation

Haoge Jiang Han Wang*
Department of Electrical and Electronic Engineering
Nanyang Technological University
Singapore, 639798
{haoge001@e.ntu.edu.sg, hw@ntu.edu.sg}

Wei-Yun Yau Kong-Wah Wan
Department of Institute for Infocomm Research
Agency for Science, Technology and Research
Singapore, 138632
{wyyau@i2r.a-star.edu.sg, kongwah@i2r.a-star.edu.sg}

***Abstract*—Conventional navigation techniques have mainly relied on a global information approach, wherein pre-built laser or camera environment maps are used to construct a path from a given start to destination. While these methods have seen much success, they are mainly confined to operate in simple and relatively static environments. Not only is substantial effort required for prior mapping, there is no ability for these navigation systems to learn and generalize to new unseen places. These related problems have motivated researchers to turn to machine learning approaches. In particular, the advent of Deep Reinforcement Learning (DRL) has shown much promises in tasks like context-awareness, navigating in dynamic environment, and map-less navigation. This paper attempts to survey some recent DRL papers, examining the underlying foundation for applying DRL to navigation, and highlighting five key limitations: (1) low sample efficiency, (2) the gap from simulation to real, (3) vulnerability to being trapped in local dead corners, (4) deficient collision avoidance in dynamic environments such as multi-pedestrian and multi-agents environments, (5) and lack of proper evaluation benchmark. We argue that these limitations must be addressed before the pervasive use of service robots in human society.**

Keywords—robot navigation, Deep Reinforcement Learning

I. INTRODUCTION

Robot navigation is a vital field in robot application, it is necessary for a robot to achieve safe autonomous moving and finally arrive at the desired target in various complex environments. Navigation can be defined as the process or activity of accurately determine one's position and planning a route to follow.

Conventional global information-based navigation has been used widely for most of the mobile robots and also get much success in the past years. Traditional methods like simultaneous localization and mapping (SLAM), using high precision laser or camera to build a prior obstacle map of environment[1], nevertheless this conventional method usually is likely to be associated with large computation [2], [3], so there are three unresolved issues: (1) Time-consuming of building the obstacle map and updating, (2) High reliance on accurate dense laser sensor for local cost map prediction and mapping, and (3) mainly confined to operate in simple and relatively static environments and hard to learn and generalize well to new unseen scenarios.

With the development of various deep learning techniques, researchers are motivated to turn to solve these above problems by using machine learning methods. A new idea has been proposed in recent years for the less addressed issues, applying the DRL method in robot navigation. As a vital category, DRL based methods have rapidly become popular and have been hugely successful in several tasks, including video games like Atari [4] and agent continuous control [5]. To date, many DRL-based methods have demonstrated that control commands can be derived efficiently from the original sensor input [6]-[10]. For mobile robots, complex environments greatly expand the sample space, while DRL methods usually extract actions from discrete Spaces to simplify the problem. The mobile robot using DRL method to perform the navigation task does not rely on the prior collection of various kinds of environmental information, with raw input of sensor, output a control command. The prior environment map is not required to build so that the computation will be reduced and be capable to well generalize the unseen scenarios.

DRL method provides a new idea for the unsolved issues of conventional navigation to some extent, same as traditional methods, it still has its unresolved limitations and imperfections: (1) Low sample efficiency, collecting training

978-1-7281-5169-4/20/$31.00 ©2020 IEEE

data for DRL method from the real world is very time-consuming and difficult because the required amount of data will become very large due to the DRL model always requires several episodes of trial and error to converge. (2) Transfer difficulty, even though agent can be trained and collect train data in simulation environment, the performance in real-world can be much different, therefore, how to reduce the sensor's perceived difference between the simulated environment and the real environment is also a difficult point. (3) The agent will easily be trapped in some challenging environment like long corridors and dead corners without the global information, (4) Collision avoidance in multi-agents or multi-pedestrians environments will become a challenge since the intentions of other agents and pedestrians are only partially observable to the robot, predicting all the potential available trajectory of others impose intensive computational demands. (5) Lack of a visual measurement for different newly proposed algorithms, all the proposed algorithms have their own superiority compared with classic algorithms, but how to compare them with each other is still lacking metrics. All these less addressed issues encourage researchers to keep working for the numerous new solutions and methods, Li Fei-Fei *et al*. creating a simulated environment to generate nearly unlimited data to do the training; Liu, M *et al*. and K. Wu *et al*. using laser or depth image to reduce the transfer difficulty; Liu, M *et al*. set both Extrinsic Reward for Map-less Navigation and Intrinsic Reward for Curiosity-driven Exploration to make robot do not be trapped in local and corner; Y. F. Chen *et al*. proposed an offline training to degrade centralized online computing into distributed offline computing enable robot can do the complicated computation avoidance in dynamic environment; Vikas Dhiman *et al.* provide a suite of metrics to try to set a proper evaluation benchmark. We will discuss more details in the later section III.

This paper survey the recent significant works of researchers whose work is applying DRL method in navigation and making effort to solve these problems. The rest of the paper is organized as follows. Section II briefly introduce the background and preliminaries. Section III provides a brief summarization of the papers of these works, their performance and difference between these works will be discussed as well. Finally, conclusions are given in Section IV.

II. BACKGROUND AND PRELIMINARIES

i. *Reinforcement Learning*

The Reinforcement Learning problem is considered as a Markov Decision Processes (MDPs), MDPs provides a mathematical framework to model random planning and decision-making problem under uncertainty [11]. An MDPs is expressed as a quintuple:

$$\mathrm{M} = (S, A, R, P, \gamma)$$

where S is a finite set of states, and A is a limited set of actions, R is the reward function which shows the real-time status-action reward signal, and the P is the transition function which represents the probability that current state *s* can transition to state *s+1* at time t+1 by taking an action *a* in state *s* at time t, $\gamma \in [0, 1]$ is the discount factor which represents the difference between future earnings and current earnings, meaning that the current reward is more important than the future feedback reward. In an MDP, a policy $\pi(a \mid s)$ describes the probability of mapping state *s* to action *a*. For given policy π, the action-value function is defined as the expectation of cumulative rewards, which can be evaluated as follows:

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi}[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,|s_0 = s, a_0 = a] \qquad (1)$$

Thus, if an action *a* is taken in state s and then the policy π is implemented, the action-value function is the expectation of the sum of the discounts. The purpose of this policy π is to enable the agent to get the maximum reward as much as possible, which can be achieved by using the Q learning algorithm, which the optimal action function is approximated by iteratively using the Bellman equation as following equation (2):

$$Q*(s_t, a_t) = R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \qquad (2)$$

ii. *Deep Q-Learning(DQN)*

Volodymyr Mnih *et al.* [4] first proposed Deep-Q-Networks in 2013 demonstrates that by this model agent can successfully learn control policies directly from high-dimensional sensory input using Deep Reinforcement Learning. They applied the method to Atari games and get the best result compared with all previous methods at that time and even surpassed human experts in three of all seven test games. It shows the power of combining deep neural networks and Q-learning. Normally, DQN consists of two deep neural networks with exactly the same structure but different parameters, an online network with parameters θ and a target network with parameters θ'. Each iteration will determine an action by using the ε-greedy policy. With probability ε select a random action a_t. Thereafter, after executing the selected random action, store the new transition in the replay memory. Simultaneously, sample random minibatch of stored transitions from the experience replay memory and then perform a gradient descent step on the following expression $\left(y_t - Q(s_t, a_t; \theta)\right)^2$ according to the loss function, y_t can be calculated as follows:

$$y_t = \begin{cases} r_t & \text{for terminal } S_{t+1} \\ r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta') & \text{otherwise} \end{cases} \qquad (3)$$

Through the process of minimizing the loss function, the online network keeps updating parameters. In the meanwhile, the parameters of the target network are fixed to generate the Temporal-Difference (TD) targets and regularly synchronized with the parameters of the online network.

iii. *Deep Deterministic Policy Gradients (DDPG)*

Deep Q-Learning is a masterpiece of deep reinforcement learning. Based on this method, many robot navigations tasks

have been implemented, but the shortcomings of the original DQN method are only applicable to discrete environments. In order to extend it to continuous environments, Lillicrap *et al.* [5] proposed deep deterministic policy gradients (DDPG) to use deep neural networks on the actor-critic reinforcement learning method. That is, the actor-critic framework. Actor executes actions. it selects action based on the currently learned strategy, which is suitable for continuous control. The critic evaluates the strategy based on the value function, and the evaluation results are used to improve the actors' strategy.

In order to make the learning process more stable and easier to convergence, DDPG method creates two copies of the neural network for the actor network (policy network) and critic network (Q network) respectively, one called online and the other called target:

$$\text{Actor network} \begin{cases} online: \mu(s|\theta^{\mu}) & gradient\ update\ \theta^{\mu} \\ target: \mu'(s|\theta^{\mu'}) & soft\ update\ \theta^{\mu'} \end{cases}$$

$$\text{Critic network} \begin{cases} online: Q(s,a|\theta^{Q}) & gradient\ update\ \theta^{Q} \\ target: Q'(s,a|\theta^{Q'}) & soft\ update\ \theta^{Q'} \end{cases}$$

After training a minibatch of data, update the parameters of the online network for both actor network and critic network through SGA (Stochastic Gradient Ascent) and SGD (Stochastic Gradient Descent) algorithm respectively, and then update the parameters of the target network, using the method of running average, the parameters of the online network are transferred to the parameters of the target network through soft update algorithm as follows:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

III. RELATED PAPERS DISCUSSION WITH ITS LIMITATIONS

i. Data inefficiency issues

DRL method has low sample efficiency issues, that is, the model needs quite a lot number of tests and mistakes before it converges. To solve these problems, Li Fei-Fei *et al*. [12] proposed a new method "Target-driven method" to achieve the faster convergence and create a simulated environment to generate countless data that can be used in agent training to solve the data inefficiency issue. There are several significant improvements listed in the paper, (1) Targets and scenes generalization, which means the targets can be found in different scenarios. And in the same scenario, different goals can be found. (2) Developed a high-quality 3D simulation environment "The House of Intelligent Reactions" (AI2-THOR). In this simulated environment, images rendered by the physical engine are entered into the deep learning framework, and the deep learning framework can send control commands according to visual input and transmit commands to the physical engine. Physical engines and learning frameworks communicate directly. So, the difference between real-world and simulation environment sensor input is fitted to a great extent so that the result in simulation can be easily transferred to real robots just by fine-tuning. The experiment result of convergence speed can well show that the proposed method "Target-driven method" has the optimal convergence compared with the baseline DRL algorithm in the same period. The main difference between Target-driven method and conventional DRL method is the parameter setting of policy. Normally, the policy always set as π ($a \mid s$) as mentioned in Section II where a is the taken action and s is the state. The main idea of this method is that first input the current-observation and target-observation to the network, then encode the two observations and get two embedding-vectors, fuse the two embedding-vectors, and finally input them into a policy-network. Training different policy-networks in different scenarios. The policy network input is a joint-representation of current-observation A and target-observation B ($a \sim \pi$ ($A, B \mid u$)). The function of policy-network is to tell the agent what action should be chosen to reach location B at location A. So the network proposed by the algorithm takes both state and target goal as the input, which generalizes the target and solves the problem that the network needs to be retrained to replace the target, even if the target has not been trained, it can also be used. Finally, the result will transfer to real robot by just fine-tuning and the result shows the speed of convergence is 44% faster than training a robot in real-world from scratch, this provides supporting evidence for simulation interactions in the real world and shows the possibility of going from simulation to real-world images with a small amount of fine-tuning.

ii. Transfer difficulty

K. Wu *et al*. [13] proposed another idea of reducing the transfer difficulty by just using a series of continuous four depth image which from four successive steps as the input of their proposed BND-DDQN Network. The network architecture can be briefly described as follow, the feature of input will be extracted through a CNN stream. Thereafter, the output vector will pass through three full connection layers, result in generating two different Q values and a state values, which work together to determine an action. The agent is trained in three different virtual environments. Environment 1 is relatively simple, environment 2 is more complex than environment 1, and environment 3 is the largest and complicated. Compared with four baseline methods DQN, DDQN, Dueling DDQN and Noisy Dueling DDQN, BND-DDQN get the largest average reward in the same training steps in all simulated environments which indicates that the algorithm achieves a satisfying performance. The result in the virtual environment finally transfers to the real robot, the robot shows the capability of safely moving in the dynamic and complex environment.

In the year 2019, they improved their work and proposed the BND*- DDQN model [14]. By using two features as input, one is a series of four consecutive depth images obtained from four consecutive steps and the other is three difference images generated from consecutive frames, the agents can learn more

valuable information more directly and make more effective and correct action choices. Moreover, besides the extrinsic reward signal, the random network distillation (RND) bonuses are regarded as intrinsic rewards during training to achieve better exploration. A β-consistency action selection strategy is also proposed to constrain the consistency in angular control command during action selection. The success rate of BND*-DDQN-RND *wl β*-consistency is over 90% in all test virtual scenarios and reached a maximum of 98% which illustrates the proposed model has a much superior generalization capability compared with the existing methods.

Liu, M *et al*. has been doing similar work [15] as well, the difference is (1) they use the laser as the main perception sensor which the 2D sensor lidar is with a negligible difference between the synthetic and actual scenarios, it is even better than vision sensor to reduce the difficulty in virtual to real transferring and (2) they use a continuous control algorithm like DDPG while K. Wu's work is based on the DQN which the action selection is in a discrete space and the goal-directed autonomous navigation task of their existing architecture is yet to be developed. A new network architecture Asynchronous DDPG(ADDPG) is also proposed, this method separates the sample collecting to another thread [16]. The test result shows parallel ADDPG is almost four times the capability of collecting data than the original DDPG and has a much faster speed of increasing Q value as well. To evaluate the performance of this algorithm, the baseline method they choose is a SLAM-based navigation method, specifically implemented by the move base package. The algorithm is based on laser scan for self-localization and obstacle map construction. The experimental results show that under the condition that the input is sparse, DRL method has a better success rate than traditional methods. The experiment also chooses a metric named Max Control Frequency: Max control frequency reflects the query efficiency of the motion planning. From the experimental results, Although the distance traveled by the DRL method is more than that by the move-base method, the running time is almost the same. It indicates that the DRL algorithm is faster in the planning process. Nevertheless, there is a problem is that when DRL algorithm meets the scene that has not been learned before, it is difficult for the agent to make good decisions whereas the traditional method has better generalization performance especially when there are various kinds of obstacles in the environment or the environment changes, so this paper is to provide a low-cost indoor robot navigation scheme to some extent, which enables agent do the navigation task by just using low-cost and low-precision sensors.

iii. Exploration of Challenge Environment

Even though using a continuous algorithm or discrete algorithm can both perform navigation tasks well as mentioned above. It is a tricky issue to encourage the agent to explore space as much as possible, that is, the agent should not trap in some challenging environment like dead corners and long corridors or always explore the same place. Liu, M *et al*. proposed an ICM model [17], previous DRL methods focused more on local obstacle avoidance or relied on global planning to provide sub-goal or primitives, but this paper combined the Intrinsic Curiosity Module, to build models, model predictions and feedback, then obtained the difference between predicted state and ground truth as Intrinsic Reward signal. Thus, when the current network ∅ makes use of the knowledge it has learned to predict the next state, the greater the prediction error, the stronger the effect of excitation. And navigation also relies on the suitable shape of extrinsic reward function to perform tasks such as reaching the target point, avoiding obstacles, distance from the target point, and so on. The virtual environment experiment result shows that the agents always tend to find shorter paths and the intrinsic motivation indeed attracts them towards novel and currently less predictable states even though they are not always reaching the target successfully and stuck in some structures which are difficult to escape because of the conflict of random exploration and intrinsic motivation.

iv. Collision Avoidance in dynamic environment

There are also many researchers' works are focus on collision avoidance in dynamic environment. Y. F. Chen *et al*. [18],[19] proposed that in the multi-agents environment where each agent cannot establish reliable communication with each other, finding an available collision-free path can be a challenge since each agent' s intention is unobservable. In the meanwhile, apart from the other agents, the agent is also expected to avoid pedestrians because there are always people around in the real scenarios, so it also needs to model human behavior and navigation rules. The point is the pedestrians or other agents in the environment are not just a moving obstacle, they are also constantly making decisions that the states, policies, and intentions of other agents and pedestrians are only partially observable to the robot. Therefore, once the agent need to plan a path that is feasible for all neighboring agents in the environment, uncertainty in environmental models and calculations can lead to inaccurate prediction planning for other agents especially the trajectory prediction after several seconds, this requires a high update frequency, which will cause an intensive computation, thus, the agent can be difficult to make the precise obstacle avoidance decision. In these two papers, all the states and inputs are transformed into the robot ontology coordinate system, and the estimated states (including position, speed, size and other information) of the own state and adjacent individuals are taken as inputs, considering the uncertainty of other individual motion, a value network for time estimation is constructed. At the same time, the latter [19] also formulated a reward function based on the robot's behavior rule (such as driving on the right). The offline training degrades centralized online computing into distributed offline computing. From the experimental results, it shows that the performance of using offline training to do robot obstacle avoidance and motion planning has achieved satisfying results, especially for complex dynamic environments, such as high-traffic campuses and exhibitions.

M. Everett *et al.* then make some improvements in his paper [20], the difference is that no longer assume the

behavioral rules of other individuals, in other words, assume that the others do not cooperate, that is, the agents do not adopt the same avoidance strategy, in the meanwhile, LSTM is added to predict the status of any other individual, this allows the algorithm to make corresponding decisions based on any number of agents around the robot. The expensive multi-line lidar is also abandoned and uses a single line lidar instead also reduced the cost of constructing a robot platform.

Another work from J. Pan *et al*. also achieved remarkable results in collision avoidance [7]. Their approach is that removes the manual control stage and do not use ROS navigation. It uses Deep Reinforcement Learning algorithm PPO to automatically collect data and formulate reward functions to achieve end-to-end mapping of laser to continuous speed and migrate to multi-robot navigation. At the same time, scene classification is performed based on environmental information, so different motion policies are adopted to improve efficiency and ensure safety.

v. Evaluate metrics for comparison

For all the above works, most of them do the evaluation by compared the proposed algorithm with the baseline methods which normally are traditional algorithms and architectures. To have a direct and clear comparison of all the algorithms, Vikas Dhiman *et al.* provide a suite of metrics to try to set a common evaluate standard. [21]. They provide two main metrics: (1) Latency-1:>1 metric, the ratio of time taken to first find the goal to the average time taken to revisit the goal. (2) Distance-inefficiency metric, the ratio of distance covered by the agent as compared to the approximate shortest path length to the goal. Thus, it can explicitly evaluate the ability of algorithms to exploit map information and capable to answer whether a representative DRL algorithm is able to exploit map information under this experimental suite.

For easy organized, a table is listed to better category the related paper's work as follows:

TABLE I. PROMINENT DRL METHOD STUDIES

Author	Sensor Category	Purpose	Method
12	Vision based	Decision making	Target-driven
13	Vision based	Decision making	BND-DDQN
14	Vision based	Decision making	BND*-DDQN
15	Laser based	Decision making	ADDPG
17	Laser based	Decision making	ICM
18	Laser based	Collision avoidance	CADRL
7	Laser based	Collision avoidance	PPO
19	Laser based	Collision avoidance	SA-CADRL
20	Laser based	Collision avoidance	GA3C-CADRL

From the table we can clearly know either Laser-based or Vision-based method both widely used to perceive the environment, Vision-based system can obtain more 3D information from the environment, whereas the laser-based system takes advantage of the negligible difference between the simulated environment and actual scenarios so the transfer from simulation to real deployment is easier. These above works use different structures base on Deep Reinforcement Learning, and all of them achieve the satisfying performance. The limitation is the data, almost of works have to train the agent in simulation first and then test in real world, so the problem of transferring result from simulation to reality is still waiting to be well solved.

IV. CONCLUSION

There has been tremendous progress in Robot navigation using various techniques. DRL method provides a new architecture and new idea to solve the less addressed issues. Though new techniques have opened up many possibilities for robot navigation, there are many complex grey areas to be looked at, most prominent being virtual to real transfer and data inefficiency. The conflict is that normally you just can get enough data to train the agents only in simulated environments, but the process of transferring performance from simulation to reality can be very inefficient and difficult. Aiming at a series of model-free RL problems, Meta RL combining Imitation Learning and Few shot Learning is also considered as one of the future solutions. Due to the vastness of this field, there have been many researches in this area which has been provided above and many are still going on, the process of perfecting the navigation process will still go on for many years to come.

ACKNOWLEDGEMENT

This research is partly supported by A*STAR grant no. 192 25 00049 from the National Robotics Programme (NRP), Singapore.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," IEEE robotics & automation magazine, vol. 13, no. 2, pp. 99–110, 2006

[2] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, “Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 1047–1054, 2017.

[3] X. Zhang, A. Liniger, and F. Borrelli, “Optimization-based collision avoidance,” arXiv preprint arXiv:1711.03449, 2017.

[4] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.

[6] F. Sadeghi and S. Levine, “Cad2rl: Real single-image flight without a single real image,” arXiv preprint arXiv:1611.04201, 2016.

[7] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, “Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning,” arXiv preprint arXiv:1709.10082, 2017.

[8] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, “Oneshot reinforcement learning for robot navigation with interactive replay,” arXiv preprint arXiv:1711.10137, 2017.

[9] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, “Reinforced imitation: Sample efficient deep reinforcement learning for map-less navigation by leveraging prior demonstrations,” arXiv preprint arXiv:1805.07095, 2018.

[10] L. Xie, S. Wang, S. Rosa, A. Markham, and N. Trigoni, “Learning with training wheels: Speeding up training with a simple controller for deep reinforcement learning.” Institute of Electrical and Electronics Engineers, 2018.

[11] Kearns, M.; Mansour, Y.; Ng, A.Y. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Mach. Learn. 2002, 49, 193–208. [CrossRef]

[12] Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J.J.; Gupta, A.; Fei-Fei, L.; Farhadi, A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3357–3364.

[13] K. Wu, M. Abolfazli Esfahani, S. Yuan, and H. Wang, “Learn to steer through deep reinforcement learning,” Sensors, vol. 18, no. 11, p. 3650, 2018.

[14] K. Wu, H. Wang, M. A. Esfahani and S. Yuan, "BND*-DDQN: Learn to Steer Autonomously through Deep Reinforcement Learning," in IEEE Transactions on Cognitive and Developmental Systems. 2019

[15] Tai, L.; Paolo, G.; and Liu, M. 2017. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. arXiv preprint arXiv:1703.00420.

[16] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” arXiv preprint arXiv:1610.00633, 2016.

[17] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, “Curiosity-driven exploration for mapless navigation with deep reinforcement learning,” arXiv preprint arXiv:1804.00456, 2018

[18] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan How. “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning.” arXiv preprint arXiv:1609.07845, 2016.

[19] Y. F. Chen, M. Everett, M. Liu, and J. P. How, “Socially Aware Motion Planning with Deep Reinforcement Learning,” arXiv:1703.08862 [cs], Mar. 2017, arXiv: 1703.08862

[20] M. Everett, Y. F. Chen, and J. P. How, “Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning,” arXiv:1805.01956 [cs], May 2018, arXiv: 1805.01956.

[21] Vikas Dhiman, Shurjo Banerjee, Brent Griffin, Jeffrey M Siskind, and Jason J Corso. A critical investigation of deep reinforcement learning for navigation. arXiv preprint arXiv:1802.02274, 2018.