Visual-Based Deep Reinforcement Learning for Mobile Robot Obstacle Avoidance Navigation

Zhiyuan Nan
Hanyang University
Department of Electrical and Electronic Engineering
Ansan, Korea
namjw@hanyang.ac.kr

Haewoon Nam
Hanyang University
Department of Electrical and Electronic Engineering
Ansan, Korea
hnam@hanyang.ac.kr

***Abstract*—To address the issue of navigation failure caused by light reflection in real-world navigation scenarios using inexpensive 2D LiDARs, traditional SAC-based algorithms face challenges such as inability to train in highly randomized and sparsely rewarded environments, as well as slow training. In this paper, we propose a combination of a monocular camera and a depth estimation model as a substitute for the inexpensive 2D LiDAR and introduce a variant algorithm called Sharing Encoder Self-Attention Soft Actor Critic (SESA-SAC) for collision-free indoor navigation of mobile robots. To improve the efficiency of robot learning in sparse environments, we collect expert data from 200 episodes and store them in a replay buffer. We conduct training by randomly sampling from both exploration data and expert data, without pre-training. To enhance training performance, we introduce a channel-wise self-attention structure and layer normalization in the network to learn better features. Additionally, we propose a shared feature extractor to achieve more stable training. Moreover, we conduct training and testing in GAZEBO, and the experimental results demonstrate that our proposed SESA-SAC algorithm outperforms traditional SAC algorithms in terms of convergence speed, stability, and efficiency for indoor navigation tasks.**

***Index Terms*—real-world, deep reinforcement learning, indoor navigation**

I. Introduction

In contemporary years, there has been a surge in the evolution and application of autonomous robotic navigation technology across a myriad of sectors, including industry, service, domesticity, agriculture, and the exploration of unfamiliar territories. Nevertheless, the algorithms deployed in these domains predominantly adhere to the conventional mode of robotic navigation such as A* [1], Dijkstra [2], and Dynamic Window Approach (DWA) [3] among others. These traditional autonomous navigation algorithms for robots often necessitate accurate environmental models and sophisticated planning mechanisms for successful implementation. Although they facilitate robotic navigation to a certain degree, they grapple with numerous unforeseen or unmodelled scenarios in real-world conditions. These include dynamic obstacles, sensor inaccuracies, complex topographies, and diverse lighting conditions, all of which can culminate in the failure of these traditional algorithms to navigate correctly. In addition, the traditional algorithms exhibit significant drawbacks when navigating unexplored terrains. In such settings, the robots are incapable of precisely modelling the environment, necessitating reliance on perception, localization, and the construction of environmental maps using methodologies such as Simultaneous Localization and Mapping (SLAM) [4]. Subsequent to this, the robots conduct path planning and autonomous navigation, hinged on these environmental models. This sequence of actions demands considerable computational resources and time, and in intricate environments, may give rise to modelling and localization inaccuracies that result in navigation failures.

Traditional reinforcement learning (RL) algorithms are computational methodologies designed to ascertain optimal control strategies. However, these conventional RL algorithms confront substantial limitations when dealing with high-dimensional data, often the kind of output produced by sensors, thereby impeding their capacity to learn efficiently from such information. In an attempt to ameliorate these drawbacks, scholars in recent years have explored the fusion of traditional RL algorithms and deep neural networks, leading to the emergence of deep reinforcement learning (DRL). This innovative approach is capable of effectively learning the mapping relationships between high-dimensional sensor features and robotic actions via deep neural networks. Furthermore, DRL does not necessitate the use of map information during navigation, circumventing the intricacy involved in manually creating environmental models and devising planning algorithms. The DRL-based navigation algorithm has become one of the key approaches to solve the autonomous navigation problem due to its excellent performance. Many recent studies on autonomous navigation based on deep reinforcement learning are based on 2D LiDAR sensors. For example, Zhou et al. [5] proposed an autonomous navigation algorithm based on LSTM-DDPG. Jia et al. [6] proposed an autonomous navigation algorithm based on GRU-Attention based TD3. Jiang et al. [7] proposed an autonomous navigation algorithm based on ITD3-CLN, etc.

However, in real-world situations where mobile robots navigate using inexpensive 2D LiDAR, collisions and navigation failures are common when faced with black or reflective obstacles. This is mainly due to the robot's inability to perceive these obstacles. Navigation with the depth camera does not present such a problem. This paper contemplates the prospective viability of large-scale deployment in real-world contexts. Given the prohibitive expense associated with

979-8-3503-1327-7/23/$31.00 ©2023 IEEE

incorporating depth cameras on robots for mass deployment, we put forth a potential alternative: the combined use of cost-effective monocular cameras and depth estimation models as substitutes for depth cameras. This amalgamation enables the perception of analogous information in both simulated and real environments. In this study, we introduce an enhanced algorithm premised on the Soft Actor-Critic framework. By bolstering the network's capacity for feature extraction and standardizing the input images, the proposed algorithm relies solely on the robot's forward-facing image data and target position details to execute end-to-end navigation and obstacle avoidance tasks. This novel algorithm can be trained in environments characterized by a high degree of stochasticity and sparse rewards, managing to regulate the robot's movement within a continuous action space. The efficiency of this improved algorithm in performing obstacle avoidance and navigation tasks is substantiated through a series of simulation experiments. The structure of the paper is as follows: The succeeding section delves into the body of work pertinent to this study. In the third segment, we elucidate the proposed enhancement of the algorithm. The fourth and fifth sections are devoted to discussing the outcomes of the conducted experiments and providing a synopsis of the research respectively.

II. Soft Actor-Critic

The Soft Actor-Critic (SAC) [8] is an off-policy, maximum entropy-based reinforcement learning algorithm, predominantly employed in the resolution of control problems within continuous action spaces. Being a stochastic policy algorithm grounded in the principle of maximum entropy, the SAC boasts superior exploratory capabilities and robustness in comparison to the Deep Deterministic Policy Gradient (DDPG) algorithm [9]. It demonstrates greater adaptability in the face of interference, facilitating smoother adjustments. An enhancement in training speed is also observed, and the incorporation of maximum entropy promotes more uniform exploration within the algorithm. Given its impressive performance, the SAC is extensively utilized in the realm of robotic control. The SAC encompasses two integral network components: the Actor and the Critic networks. The latter comprises dual action-value function networks. The loss function of the actor network is defined as

$$\mathcal{L}_\pi(\phi) = \mathbb{E}\Big[\alpha \log(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\Big] \tag{1}$$

Where s_t is selected from the replay buffer, a_t is determined by the actor, and α is a tunable temperature coefficient hyperparameter. The update of the target action value is defined as

$$\begin{aligned} Q_{\text{target}}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \Big[\min_{i=1,2} Q_{\theta_i'}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) \\ - \alpha \log\left(\pi_\phi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right)\Big] \end{aligned} \tag{2}$$

Where the $r(\mathbf{s}_t, \mathbf{a}_t)$ and s_{t+1} are obtained from the replay buffer, a_{t+1} is determined by the actor, and $Q_{\theta_i'}$ represents the target action value network. The loss function of the critic network is defined as

$$\mathcal{L}_{\text{q}}(\theta) = \mathbb{E}\Big[\big(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - Q_{\text{target}}(\mathbf{s}_t, \mathbf{a}_t)\big)^2\Big] \tag{3}$$

Where Q_θ represents the predicted action value network.

III. Proposed Method

This paper presents a proposal for a visually-guided mobile robotic obstacle avoidance navigation system that is underpinned by the Sharing Encoder Self-Attention Soft Actor-Critic (SESA-SAC) algorithm. By integrating a self-attention network structure, the depth map feature extraction network has the capacity to distill more beneficial features, thereby accelerating the efficacy of training in the context of deep reinforcement learning. The objective of this method is to augment the capacity of mobile robots to navigate and circumvent obstacles using exclusively visual information, accomplished through the learning of superior features.

A. Problem Definition

We use Markov Decision Process (MDP) [10] to define the problem of mobile robot navigation. Firstly, MDP is composed of a quintuple M=(*S*, *A*, *R*, *P*, γ). *S* represents the set of states, *A* represents the set of actions in the decision process, *R* represents the reward function, where *r* (s_t, a_t) represents the immediate reward obtained by performing action $a_t \in A$ in state $s_t \in S$. *P* represents the state transition matrix. γ represents the discount factor, with $\gamma \in [0, 1]$. Since we need to use MDP to solve the problem of mobile robot navigation, we will set the state space *S*, action space *A*, and reward function *R* separately.

1) State space S: Given that a solitary depth map possesses restricted information, we have configured our state setting to not only perceive the contemporary depth map, but also amalgamate information derived from antecedent depth maps. This amalgamation engenders a sequential state input akin to short-term memory. Utilizing an unoccupied array, we accumulate the prediction depth maps from I_{t-3} to I_t. The resultant input state space is defined as

$$S_t = \{I_{t-3}, I_{t-2}, I_{t-1}, I_t\}, \tag{4}$$

where I_t represents the current depth map state, and I_{t-n} represents the previously observed depth map state by the robot. with $n \in \{1, 2, 3\}$

In order to achieve successful navigation, The position information of the mobile robot and the navigation target point is defined as

$$P_t = \{d_t, \theta_t, v_{t-1}, \omega_{t-1}\}, \tag{5}$$

where d_t is the distance between the mobile robot and the navigation target point, θ_t is the angle between the forward direction of the mobile robot and the navigation target point, v_{t-1} is the linear velocity action previously taken by the robot, and ω_{t-1} is the angular velocity action previously taken by the robot.

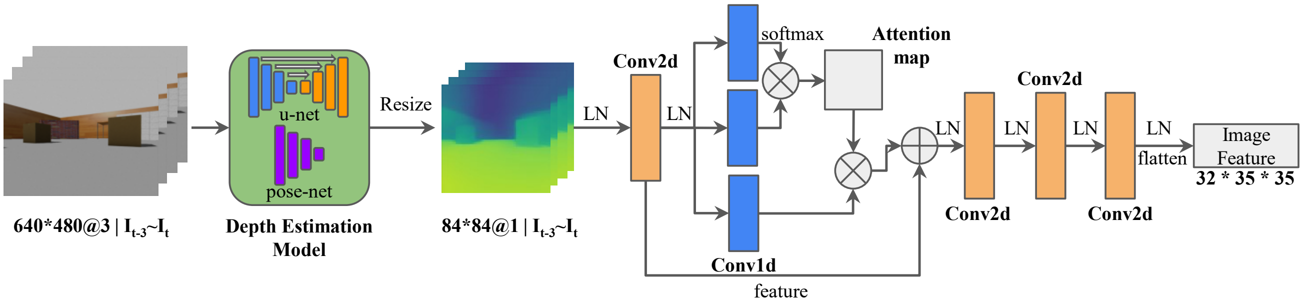

Fig. 1. Feature extraction network architecture.

2) Action space A: The mobile robot has two actions: linear velocity and angular velocity. The continuous action space is defined as

$$A_t = \{v_t,\ \omega_t\}, \tag{6}$$

where v_t represents the current linear velocity of the robot, with v_t ∈ [0, 0.22], and ω_t represents the current angular velocity of the robot, with ω_t ∈ [-1, 1].

3) Reward function R: We define a sparse reward function as

$$r(s_t, a_t) = \begin{cases} \alpha * v - \lambda * |\omega| - \delta * d_o, & \\ r_g & d_t < d_\beta, \\ r_c & if \quad collision, \end{cases} \tag{7}$$

where d_o is the minimum distance between the robot and obstacles. If d_o is less than 1, the calculation is done using the function 1-d_o. Otherwise, d_o is 0. In order to encourage exploration by the mobile robot and avoid situations where the robot rotates in place, the absolute difference between the linear velocity and angular velocity is set as the reward value. α, λ, and δ are hyperparameters set to 0.5 in this paper. When the distance between the robot and the navigation target point is less than the set d_β value, it is considered as reaching the target point and positive reward is given. If a collision occurs, negative reward is given. In this paper, d_β is set to 0.2.

B. Network architecture

As depicted in Fig. 1, we put forth a unique network architecture, introducing a Channel-Wise Self-Attention (CWSA) network [**?**] structure subsequent to the initial convolutional layer in the feature extraction network. Moreover, we incorporate Layer Normalization (LN) layers [11] at the input and intermediate junctures of the network. Considering that our input states are constituted by four sequential depth maps, the addition of the CWSA structure facilitates a more efficacious extraction of pertinent feature information from the interaction amongst the channels. CWSA is a self-attention mechanism that operates individually on each channel or feature map within a neural network, focusing on modelling the interdependencies amongst channels within a feature map. This enables the network to learn channel-specific attention weights, emphasizing the significance of various channels during the processing of input data. By taking into account the relationships between channels, the network can effectively seize global dependencies and augment its proficiency in extracting relevant information from the input. The application of LN ensures that the outputs of the intermediate layers of the neural network maintain a uniform mean and variance, thereby enhancing their numerical stability. This contributes to accelerating the convergence of the neural network.

The comprehensive training procedure for the network commences with the collection of RGB images via the robot's front-facing camera. These RGB images are subsequently transmuted into depth maps utilizing a Depth Estimation Model. The depth maps are accumulated, thereby producing a sequence of four successive predicted depth maps that serve as the input state. This input state undergoes normalization and is then introduced into the feature extraction network. Following a passage through a Layer Normalization (LN) layer, the input state is incorporated into the initial convolutional network layer. The output features derived from this layer traverse another LN layer and are further manipulated by the Channel-Wise Self-Attention (CWSA) structure, resulting in the extraction of more intricate features. These distilled features traverse another LN layer and are inputted into the subsequent convolutional layer. This procedure is replicated until the ultimate state features are extracted. All convolutional layers, barring the CWSA structure, employ the Rectified Linear Unit (ReLU) activation function.

After extracting feature information through the feature extraction network, the features are inputted into the actor-critic network structure as depicted in Fig. 2. We share a single feature extraction network between the actor and critic networks. In the actor network, the features extracted by the feature extraction network are further compressed and extracted through a fully connected layer, then concatenated with position information features also extracted from a fully connected layer. The concatenated features are inputted into two additional fully connected layers, and in the end, the mean and variance related to the action are outputted. The linear and angular velocity actions under the current state are calculated using the reparameterization technique. In the critic network, the features extracted through the feature extraction network

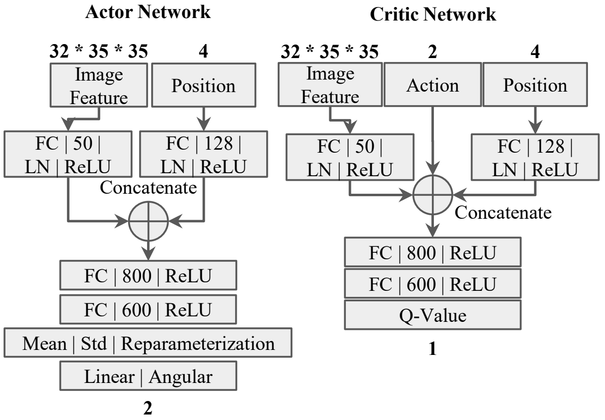

Fig. 2. Actor-Critic network architecture.

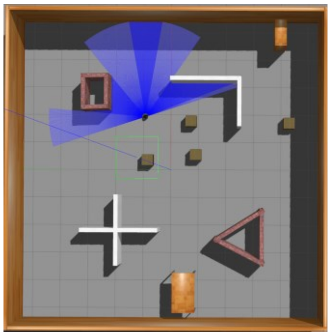

Fig. 3. Training environment.

are further compressed and extracted via a fully connected layer. These features are then concatenated with the current action and position information features. The concatenated features are input into two additional fully connected layers to predict the Q-value associated with the current action in the current state. During the updating process, only the critic network is used to update the parameters of the feature extraction network.

Within the feature extraction network, four convolutional layers are defined with the following parameters: the first layer has an input size of 3, an output size of 32, a kernel size of 3, and a stride of 2; the subsequent three layers all possess an input size of 32, an output size of 32, a kernel size of 3, and a stride of 1. A fully connected layer, consisting of 128 neurons and employing the ReLU activation function, is designed for position information extraction. In the actor-critic network, two fully connected layers are incorporated for the extraction of features and prediction. These layers are characterized by 800 and 600 neurons respectively, both utilizing the ReLU activation function.

IV. Experimental Results

A. Experimental Environment

The efficacy of the proposed algorithm is assessed in this study through conducting obstacle avoidance and navigation trials using a mobile robot in GAZEBO. Three distinct experiment sets were performed: the traditional SAC algorithm, the traditional SAC algorithm supplemented with input normalization, and the proposed SESA-SAC algorithm. The traditional SAC algorithm utilized a four-layer convolutional network, with parameters aligning with those applied in the convolutional layers of the SESA-SAC. Nonetheless, in the traditional SAC algorithm, the position information in the Actor-Critic network was not subjected to feature extraction; it was directly concatenated with the extracted sequential depth map features. The remaining parameters for both the traditional SAC and the proposed SESA-SAC were identical, as illustrated in Table I. The experimental setup comprised an NVIDIA RTX 2070 SUPER GPU, Robot Operating System (ROS: Melodic), and the GAZEBO simulator. The robot's training and testing were performed within the GAZEBO simulator. For the purpose of the experiments, the turtlebot3 burger robot was deployed in the test environment, as depicted in Fig.3.

TABLE I
Parameters

	SAC	SESA-SAC
learning rate	3e-4	3e-4
gamma	0.99	0.99
tau	5e-3	5e-3
log std min	-20	-20
log std max	2	2
buffer size	100000	100000
batch size	128	128

The mobile robot obtains the current environmental RGB image through its front-mounted camera. Subsequently, a depth estimation model is employed to convert this RGB image into a predicted depth map. The deep reinforcement learning network's input state is a sequence comprised of four consecutive predicted depth maps. The DistDepth network [12], which is pre-trained, serves as our depth estimation model, offering precise predictions of depth maps derived from singular RGB images. Given our usage of a sparse reward function, we accelerate training by initially accumulating data via an expert agent across 200 episodes and storing this in a replay buffer. However, the robot was not subjected to pre-training. Rather, following data gathering through the expert agent, the agent that necessitates training persists in exploring the environment with its original policy, collecting additional data. Training then proceeds through random samples obtained from the replay buffer. Both the traditional SAC algorithm and the algorithm proposed within this study adhere to identical training methodologies. The robot, within its environment, is required to discern obstacles using visual information, and learn to successfully navigate towards the target point while evading these obstacles. In instances where the robot collides with an obstacle during training, the positions of the robot, the obstacle, and the target point are all randomly reset, after which the robot recommences exploration. With an increase in collected data and training iterations, the robot gradually

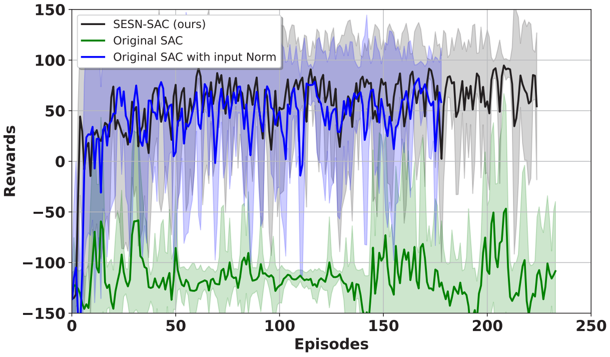

Fig. 4. Train results.

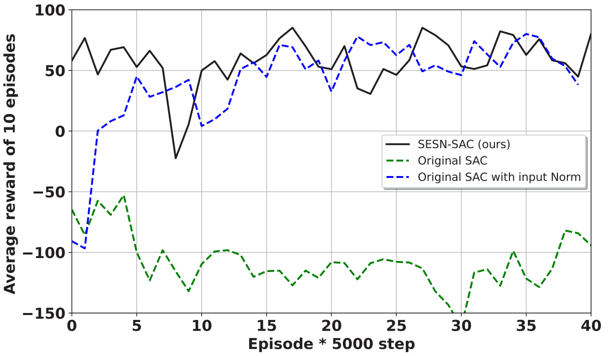

Fig. 5. Test results.

learns to circumnavigate obstacles and navigate successfully to the target point.

B. Experimental Result

In the training setup, the maximum number of steps per episode is set to 500. If the episode exceeds 500 steps, the environment is reset.

As shown in Fig. 4, this is the result graph obtained during the training process, with the results recorded after each episode. For performance comparison, the training results in the graph represent the average reward value every 10 episodes. From the training result graph, we can see that the traditional SAC algorithm without input normalization fails to train successfully. The traditional SAC algorithm with input normalization, on the other hand, achieves successful training but exhibits significant fluctuations in the obtained reward values. In contrast, our proposed SESA-SAC algorithm demonstrates greater stability compared to the traditional SAC algorithm with input normalization and achieves higher reward values.

As shown in Fig. 5, this is the result graph obtained during the testing process. We conducted testing by performing 10 episodes every 5,000 steps, and the test results represent the average value of those 10 episodes. From the testing result graph, we can observe that our proposed SESA-SAC algorithm achieves a reward value of 50 or above right from the beginning of training. The traditional SAC algorithm, on the other hand, fails to train successfully. The traditional SAC algorithm with input normalization starts to converge and reaches a reward value of 50 or above at around 25,000 steps.The proposed algorithm achieved an average reward value of 58.08, while the traditional SAC algorithm resulted in -112.62. The traditional SAC algorithm with input normalization yielded a value of 41.72.

V. Conclusion

This study introduces a novel algorithm, Sharing Encoder Self-Attention Soft Actor Critic (SESA-SAC), based on Soft Actor Critic (SAC), designed for application in obstacle avoidance navigation systems. The algorithm enhances its capacity for feature extraction through the integration of a Channel-Wise Self-Attention (CWSA) structure within the feature extraction network and the implementation of layer normalization at the neural network's input and output stages. A shared feature extraction network is utilized by the actor and critic networks, with the critic network tasked with updating the feature extraction network, contributing to an efficient and stable training process. This novel methodology successfully navigates training in environments characterized by extensive randomization and sparse reward distribution. The validation of the proposed approach is accomplished experimentally within the GAZEBO simulator, utilizing identical parameters and training environments as those employed by the traditional SAC algorithm. The results conclusively demonstrate that our proposed approach surpasses both the traditional SAC algorithm and other variant algorithms, achieving the task of robot obstacle avoidance navigation with commendable efficiency.

Acknowledgment

This work was supported by the Technology development Program (No. RS-2022-00164803) funded by the Ministry of SMEs and Startups (MSS, Korea).

References

[1] G. Nannicini, D. Delling, L. Liberti, and D. Schultes, "Bidirectional A Search for Time-Dependent Fast Paths," in *Experimental Algorithms*, ser. Lecture Notes in Computer Science, C. C. McGeoch, Ed. Berlin, Heidelberg: Springer, 2008, pp. 334–346.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[3] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar. 1997, conference Name: IEEE Robotics & Automation Magazine.

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

[5] Q. Zhou, L. Lyu, and H. Liu, "Deep Reinforcement Learning with Long-Time Memory Capability for Robot Mapless Navigation," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, May 2022, pp. 1215–1220.

[6] J. Jia, X. Xing, and D. E. Chang, "Gru-attention based td3 network for mobile robot navigation," in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2022, pp. 1642–1647.

[7] H. Jiang, M. A. Esfahani, K. Wu, K.-w. Wan, K.-k. Heng, H. Wang, and X. Jiang, “iTD3-CLN: Learn to navigate in dynamic scene through Deep Reinforcement Learning,” *Neurocomputing*, vol. 503, pp. 118–128, Sep. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222008347

[8] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft Actor-Critic Algorithms and Applications,” Jan. 2019, arXiv:1812.05905 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1812.05905

[9] 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,” Jul. 2019, arXiv:1509.02971 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1509.02971

[10] M. L. Puterman, “Chapter 8 Markov decision processes,” in *Handbooks in Operations Research and Management Science*, ser. Stochastic Models. Elsevier, Jan. 1990, vol. 2, pp. 331–434. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705801720

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization,” Jul. 2016, arXiv:1607.06450 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1607.06450

[12] C.-Y. Wu, J. Wang, M. Hall, U. Neumann, and S. Su, “Toward Practical Monocular Indoor Depth Estimation,” Mar. 2022, arXiv:2112.02306 [cs]. [Online]. Available: http://arxiv.org/abs/2112.02306