Exploring Multi-View Perspectives on Deep Reinforcement Learning Agents for Embodied Object Navigation in Virtual Home Environments

Xiaotian Liu
liu.x@queensu.ca
Queen's University
Kingston, Ontario, Canada

Victoria Armstrong
victoria.armstrong@queensu.ca
Queen's University
Kingston, Ontario, Canada

Sara Nabil
sara.nabil@queensu.ca
Queen's University
Kingston, Ontario, Canada

Christian Muise
christian.muise@queensu.ca
Queen's University
Kingston, Ontario, Canada

ABSTRACT

Recent years have brought the exploration of embodied reinforcement learning agents in a variety of domains. One of the advantages of artificial agents is that they can obtain visual inputs simultaneously using multiple input devices. This work explores multi-view reinforcement learning for object navigation tasks in 3D rendered virtual home environments using AI2-THOR. We trained CNN based Deep Q-learning embodied agents with egocentric, allocentric, and combined egocentric-allocentric perspectives to locate an object in an unknown environment. We compared the results of the three RL agents, and evaluated them by both reward improvement rate, and reward obtained. We demonstrate that the egocentric perspective allows for faster reward accumulation in the earlier episodes, whereas the allocentric agents obtained better long-term rewards. Interesting results arise from the combined allocentric and egocentric perspective, where we found that the agent had the best overall results by harnessing the benefits of each perspective. The results show that while single perspective embodied agents each have their own advantages, combining both inputs yields the best overall reward. Our findings provide a foundation and benchmark for building embodied RL agents with multi-view perspectives.

CCS CONCEPTS

• **Computing methodologies** → *Artificial Intelligence, Machine learning*.

KEYWORDS

Embodied Agent, Object Navigation, Computer Vision, Deep Reinforcement Learning, Multi-View Reinforcement Learning

CASCON '21, Nov 22–26, 2021,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

ACM Reference Format:
Xiaotian Liu, Victoria Armstrong, Sara Nabil, and Christian Muise. 2021. Exploring Multi-View Perspectives on Deep Reinforcement Learning Agents for Embodied Object Navigation in Virtual Home Environments. In *Proceedings of CASCON '21: CASCON X EVOKE (CASCON '21).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

An important area of artificial intelligence research is the use of embodied agents that interact with the environment. While the environment can be real or simulated, the basic premise remains the same: the embodied agent is present in the environment and acts within it, gaining knowledge of the world in which it resides. Embodied agents perform three main types of navigation tasks: visual exploration, visual navigation, and embodied Q&A [5]. The task of visual navigation can be further decomposed in point navigation, object navigation, navigation with priors, and vision-and-language navigation [5]. We focus specifically on the task of object navigation, where the embodied agent is required to find a labeled object in a new environment.

With the recent advancements in deep learning, deep reinforcement learning techniques are being applied to embodied agents for object navigation tasks [2]. Reinforcement learning (RL) agents learn to discover objects in an environment by performing a sequence of actions that maximize the expected reward. Deep learning based RL agents often require a tremendous amount of generated samples, making them impractical in many real-life settings [19]. Thus, sample efficiency becomes an increasingly important issue for embodied RL agents.

Traditionally, only a single perspective of the state space is passed to the RL agent. However, drawing on more than one perspective of the environment should add additional information for the agent to act upon. Consider the example of driving a car - the driver's perspective shows immediate actions and obstacles, whereas road maps provide information about the current location and the destination. For embodied agents, it is often the case visual inputs from multiple cameras are available. Thus RL agents should be designed to process information from these different perspectives. Very few works have tackled multi-view reinforcement learning tasks. Recently, Li et al. proposed a framework for multi-view RL

via cross-view policy transfer [10]. Chen. et al. combination of multiple views from via different vantage points for robotics control [4]. As far as we are aware, there has not been any work that tackles multi-view RL for embodied object navigation.

In this work, we want to investigate whether a deep RL agent can effectively combine multiple visual inputs of an environment for the object navigation task. We also want to benchmark the multi-view agent against RL agents trained solely on the *egocentric* or the *allocentric* perspective. We define egocentric as the first-person perspective captured by a camera mounted the agent and allocentric as a top-down view that captures the entire scene [8]. Our object navigation experiment are virtual home environments that are rendered virtually using environment AI2-THOR [9]. We task the agents with locating an object in a living room under different levels of difficulty. We used the standard Deep Q-Learning (DQN) agent based on the convolutional neural network (CNN) [13]. To combine two different views, we constructed a dual input CNN by concatenating the latent representation of two identical CNN subnetworks. To keep network architecture consistent, each of the CNN subnetworks is identical to the CNN used in single perspective CNN.

Our experiments demonstrate that CNN-based DQN agents can effectively capture information from both perspectives. While learning from a solely allocentric or egocentric perspective has different advantages, combining both inputs yields a superior performance to either individually. Our work makes two main contributions:

- First, we compare the performance of egocentric versus allocentric DQN agents for embodied object navigation.
- Secondly, we show that combining multiple visual inputs in the latent space is advantageous for embodied deep RL agents.

2 BACKGROUND

Our work involves aspects of deep RL, object navigation, and embodied agent design. In this section, we outline the background information for key concepts discussed in later sections.

2.1 Object Navigation

Most classic navigation methods focus on using geometric measures to find an optimal path to a goal. Object navigation is a subset of the semantic navigation problem where an agent needs to discover objects through their relationships within an environment [5]. Object navigation requires both spatial and conceptual information. Embodied agents and human beings store information differently. The inner working of the human brain remains mostly a mystery. However, the representation of both spatial and semantic information appears to be entangled in the brain [22]. The key to object navigation is to build a system that can bridge the gap between spatial and semantic information [7]. In this work, we focus on how to formulate object navigation as an RL problem in a virtual home environment.

2.2 Perspective and Views

In spatial representations, how relationships between objects and the viewer are defined provides different information encodings. The egocentric perspective relates objects in the scene with the agent itself [8]. In the allocentric perspective, object locations are represented in terms of each other, independent of the agent's location [8]. Specifically relating to agents for object navigation, the egocentric perspective is important as it encodes the immediate actions available to the agent. Conversely, the allocentric view provides a spatial mapping of the environment, and is more suitable for long-term planning. In computer vision, state-spaces can be represented from different vantage points. While the use of a single vantage perspective is considered single-view, combining various vantage points is called multi-view [10]. The use of multi-view reinforcement learning has been explored in relation to DQNs in the context of robot arm manipulation [4] and Atari games [10].

2.3 Deep Q-Networks

Deep Q-Networks (DQNs) are RL agents that approximate the optimal Q-Value function [15] using deep neural networks[13]. The original DQN network was able to achieve super human level performance in Atari 2600 games. To estimate the Q-Value function, the DQN is trained using experience replay, which stores a series of N previous transitions. The update equation for Q-Values (which we build on later) is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

Since the conception of DQNs, their successes have been built upon, and applied to a wide variety of problems across varying domains. DQNs have been used in business for stock market portfolio management [6], for cloud computing workflow management [17], and in the energy sector for HVAC system balancing [1]. DQNs have also been applied to the field of vision for unmanned aerial vehicle (UAV) obstacle navigation in three dimensions [18] and object navigation in indoor scenes [23].

2.4 AI2-THOR

We use AI2-THOR to setup our experiments. AI2-THOR is developed by the Allen Institute research for embodied agent research [9]. It renders realistic home environments that can be used for vision based object navigation. AI2-THOR supports as various home environments such as kitchen, living room, bedrooms and bathrooms. The agents can perform both discrete and continuous action with the option of adding stochasticity. AI2-THOR is mostly used for testing emobodied agents for tasks such as object navigation and instruction following. See *https://github.com/allenai/ai2thor*.

3 PROBLEM STATEMENT

Embodied agents can gather information in a multi-modal and multi-sensory fashion. Visual inputs from different perspectives can be easily gathered through multiple cameras. For example, when designing a home assistant robot, we can have cameras mounted in a fixed location to observe the entire room and a mobile one the agent itself. We call the fix-angle perspective the allocentric view and the mobile perspective the egocentric view. The natural question arises whether an embodied agent trained using a deep reinforcement learning framework can take advantage of both information. Thus, our experiments revolve around two search questions.

RQ1: What are the performance differences between an egocentric RL agent and an allocentric RL agent.

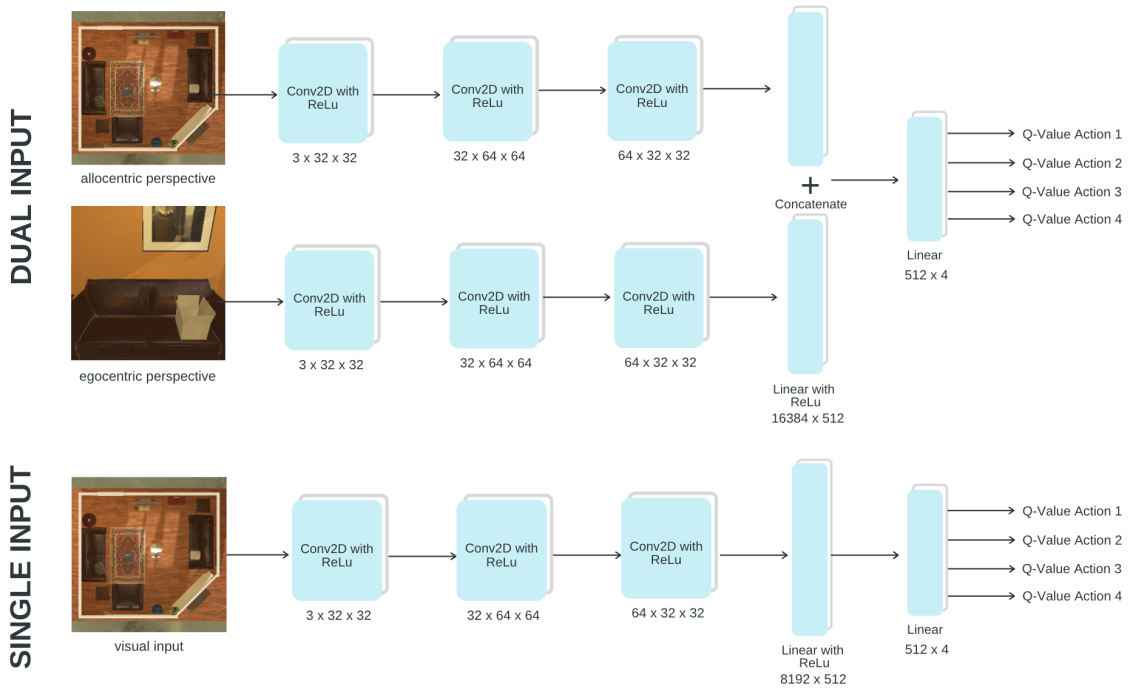


Figure 1: DQN architecture used to approximate the Q-value function for both dual (top) and single (bottom) perspective.

RQ2: Can an RL agent take advantage of both egocentric and allocentric information from two separate visual inputs.

4 METHODOLOGY

We used a CNN-based DQN network to train our embodied agents. The architecture is similar to [13] which uses a CNN DQN framework for atari games. For the multi-view agent, we concatenated the latent space of two CNNs before the final prediction layer. Our agents are trained for object navigation tasks in the AI2-THOR environment that renders the photo-realistic indoor scenes for embodied agents. The performance of the agents is evaluated based on both reward improvement rate and and the optimal policy reward. The details are outlined in this section.

4.1 Single and Dual Input CNN

We use a CNN neural network to approximate our Q-Value function. The input is a three-channel RGB image with dimensions 256 by 256. We use three convolutional layers to downsample the image into a latent representation. The first convolution layer has 32 channels with a kernel size of 8 and a stride of 4. The second convolution layer has 64 channels with a kernel size of 4 and a stride of 2. The last layer has 32 channels with a kernel size of 3 and a stride of 2. We then flatten this layer, and connect it to a fully connected linear layer of dimension 512. The ReLU activation function was used for all layers except the final output. The output layer has four dimensions which represent Q-Values for MoveForward, MoveBackward, TurnLeft, and TurnRight. We use this architecture for both the egocentric and allocentric agents.

We use a similar architecture for our dual input network. The dual input architecture has two separate single-channel CNNs to downsample both images. However, we stack the latent representation of both single-channel CNNs together before passing to the final layer. Thus, the network should weigh both input information equally by default. We chose to keep the architectures of both networks as similar as possible to facilitate a comparison between the information gained using a single-view compared to a multi-view.

4.2 Deep-Q Learning

We followed the standard Deep Q-learning framework as outlined in [11]. In order to use our dual CNN, we modified the Q-learning algorithm with our novel network architecture. Our method is trained using raw images generated from the AI2-THOR. The idea is to calculate the Q-value from given images directly using a deep neural network. For each (s_t, a_t, r_t, s_{t+1}) tuple generated, we update the Q-value with a learning rate $a = 0.5$ using the Bellman Equation. To update our sample independently, we implemented experience replay with a Deque memory structure of size 5000. For every 10 steps, we sample a batch of 64 images to backpropagate our neural network with a learning rate of 0.001. The agent acts in a $\epsilon - greedy$ policy with a decaying exploration rate. The randomness of sampling ensures that updates are not correlated with the current action thus avoiding local minima. A detailed description of our Deep Q-learning algorithm is shown in Algorithm 1.

Algorithm 1: Deep Q-learning algorithm

Initialize weights with replay buffer $\mathcal{D}$
repeat (for each episode)
  for each step **do**
    Observe state s_t and select $a_t \sim \pi(a_t, s_t)$
    Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
  for each sample **do**
    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Target Q-value:
      $Q^*(s_t, a_t) = r_t + \gamma Q_\theta(s_{t+1}, \max_{a'} Q_\theta(s_{t+1}, a'))$
    Gradient descent on $L(Q^*(s_t, a_t), Q_\theta(s_t, a_t))$
  until s is terminal

Figure 2: Easy Scenario, left, Hard Scenario, right

5 EXPERIMENTS

We designed three separate agents, each with a different view of the environment. The egocentric agent takes the first-person view of the environment, while the allocentric agent takes the top-down view of the environment. These two agents use the same CNN-based DQN architecture. The dual-perspective agent uses our novel dual-input CNN architecture. It takes both the allocentric and egocentric images as input for policy learning. We evaluate the performances of these three agents based on reward improvement rate and optimal policy reward. Simple and hard tasks are evaluated separately.

We trained our agents for 100 episodes for the easy scenario and 200 episodes for the hard scenario. We conducted our experiment on Ubuntu 20.04 with an Intel i7 CPU and an Nvidia RTX2070 8GB GPU. Each episode takes approximately 5 minutes to train, with a combined training time of 75 hours.

5.1 Task Setup

To answer our research questions, we set up an RL environment for an embodied object navigation task. The objective of our RL agents is to navigate and find an object in an unknown environment. We chose a cardboard box as the target object and in a living-room setting. To test our methods, we designed two scenarios to train our RL agents. We call the first scenario the easy scenario where the box is relatively close to the agent. The optimal policy only requires 32 actions to reach the box. The agent needs to learn how to turn and navigate to the box directly. The second scenario is the hard scenario where the box is relatively difficult to discover. The optimal policy for this scenario is 65 actions. The agent needs to learn to navigate towards a target while avoiding collision with obstacles. The top-down view of both scenarios can be seen in 2.

5.2 RL Settings

Our agent was trained in a 3D living room environment in AI2-THOR. Since AI2-THOR does not have a native RL library, we had to define and implement action space and reward functions from scratch. The state of the environment is captured by both the egocentric image and the allocentric image. Both images are a 256 by 256 RGB array generated through the THOR API. To simplify our action space, we discretized our agent's action space into four types: MoveAhead, MoveBack, TurnLeft, and TurnRight. For the two movement actions, we set the distance to 0.25m for each step. For TurnLeft and TurnRight we set the rotation angle to 90°. Thus, each physical coordinate location in the environment will have four pairs of images available to the agent.

We designed the reward function to encourage exploration while avoiding obstacles. When our agent conducts a turning movement, or the action MoveBack, the agent gets a reward of -1. This is designed to discourage unnecessary movements by the agent. For the MoveAhead action, we give the agent a neutral award of +0. We designed it to encourage the agent to explore the environment while moving constantly. When the agent has collided with the wall or an obstacle during training, we penalize the agent -10. The negative award is used to help the agent to recognize obstacles and dead-ends. Finally, when the agent has found the target object, we gave it a large +100 reward. We define the successful termination condition as when the agent is 0.5m away from the target object while facing the object directly. A budget of 1000 movements is given to the agent. When this is exhausted without finding the target object, we terminate the episode for a new start.

5.3 Evaluation Criteria

Our agents are evaluated based on two criteria: the *reward improvement rate* and the *optimal policy reward*. We define reward improvement rate as the average number of episodes the agent needs to achieve a particular reward. For example, it takes agent one 20 episodes to achieve an average reward and 40 episodes for agent two. We can say that agent one has a faster reward improvement compared to agent two. The maximum reward in our setup is 100, and we split the rewards by an interval of 20. The optimal policy reward is the highest average reward for each agent given a set number of training episodes. We use a moving average of size five to smooth out our reward calculation.

6 RESULTS AND DISCUSSION

We compared average reward and episode length for all three agents for each of the two settings. Table 1a and 1b shows the reward improvement rate for each agent. Since episodic rewards depends on number actions taken, thus we use a range of -60 to 60 with an interval of 30 for the hard setting, and 0 to 60 with and integral of 20 for the easy setting. The first row of the two tables shows the reward interval, and the rest of the rows show the number of episodes needed. Final rewards are listed in the last column. Figure 3a and 3b show a moving average reward for every 20 episodes trained. These graphs demonstrate how each agent improves their

Reward Interval	-60	-30	0	30	60	Final-Rwd
Ego-Episodes	84	102	109	133	NA	59.97
Allo-Episodes	103	111	118	132	167	67.58
Comb-Episodes	86	104	110	125	178	67.04

(a) Easy setting.

Reward Interval	0	20	40	60	Final-Rwd
Ego-Episodes	27	34	47	91	62.55
Allo-Episodes	23	30	41	64	71.94
Comb-Episodes	12	15	21	32	84.16

(b) Hard setting.

Table 1: Episodes required for reward.

policy over time. We run each setting 5 times and the average rewards across them are used for comparison.

6.1 RQ1. Allocentric vs Egocentric Comparison

In the easy setting, allocentric agents seem to have a performance advantage in both reward improvement rate and optimal rewards comparing with egocentric agent. This setting is easy enough where both agents can achieve a reward of 40 within 50 episodes trained. Overall the performances of both agents are comparable to another. In the hard setting, the advantages of each agent become more apparent. The egocentric agent has a faster reward improvement rate than the allocentric agent in earlier episodes. It takes 84 episodes for allocentric agents to have a reward higher than -60 while taking 103 episodes for the egocentric agent. However, this speed advantage disappears when more episodes are trained. After 200 episodes, the allocentric agent achieves an optimal reward of 67.58 comparing to the egocentric agent, which achieves an optimal reward of 59.97. In addition, we qualitatively analyzed trajectories for each agent. We believe that egocentric is better at avoiding collision with obstacles such as chairs and tables, which gives a large negative reward. Allocentric agents have complete information about the map, thus, can obtain a better long-term policy. These results answer our first research question regarding advantage of egocentric versus allocentric agents.

6.2 RQ2. Multi-View Agent Performances

In the easy setting, the multi-view agent performs significantly better than the other two agents. We believe this is due to the extra information given to the multi-view agent, where it can locate its current state more easily. The easy setting only requires a small number of actions making the extra information much more useful. In the hard setting, our result indicates that the multi-view RL agent has the advantage of both networks. It has a similar early reward improvement rate advantage as the egocentric agent while having the same high long-term performance as the allocentric agent. The multi-vew agent achieved reward above -60 in 84 episodes and achieved a final reward of 67.04. Our results indicate that our dual network architecture can effectively leverage information from both perspectives. The latent concatenation in our dual CNN architecture seems sufficient for the agent to utilize both perspectives without additional priors. A qualitative analysis also shows that the multi-view agent learns to avoid obstacles in the same manner as the egocentric agent while achieving almost identical optimal reward as the allocentric agent. These findings confirm our assumption that an CNN based RL agent can combine information form multiple perspectives.

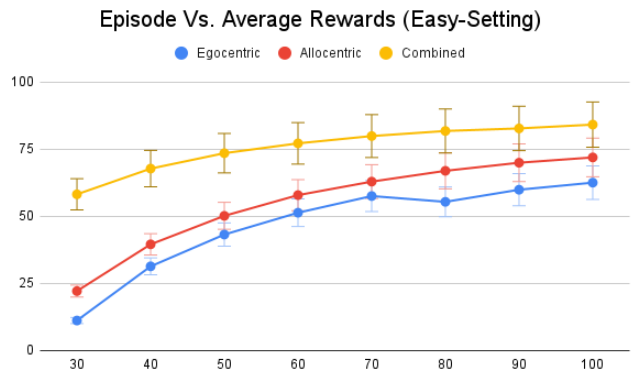
7 RELATED WORK

Most RL algorithms only use visual inputs from a single perspective. Thus very few works have tackled multi-view RL problems and their applications. Chen et al. created a model to approximate the Q-values from 4 images of a robotic arm to control its movement in the x, y, and z directions [4]. They demonstrated that multi-view inputs can be used to improve learned policies in robotic control. Li et al. examined multi-view reinforcement learning problems by extending the partially observable Markov decision processes (POMDPs) through cross-view policy transfer [10]. They showed that this extension reduces sample complexity and computational time using multi-views in video games such as Pong. Although both works tackle different aspects of multi-view RL problems, we are the first work that exams multi-view RL in embodied agents. In addition, our work is the first one to empirically compare the performance of multi-view DQN agents against agents trained solely on egocentric or allocentric input.

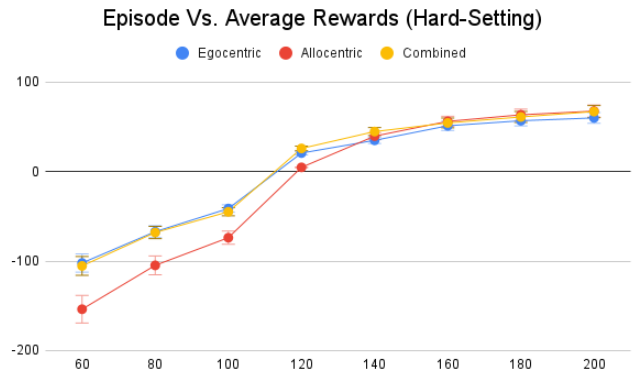
We use object navigation problems to evaluate multi-view embodied agents. Unlike our approach, works in this area mostly focus on a single egocentric perspective as input for an agent. Wijmans et al. [19] have used the end-to-end RL method to achieve state-of-the-art in point-goal navigation. Parisotto et al. [14] demonstrated that having a structured memory in RL agents helps with data efficiency. Using similar ideas, Tamar et.al [16] proposed the Value Iteration Network to solve path planning problems on a local level. Bhatti et al. [3] used a learned SLAM map to play Doom with RL agents. However, a fixed size tensor is not the only way to store memory. Wu et al. [20] used a generative model, inspired by Generative Adversarial Networks (GANs), to learn a latent representation of the environment as a graph neural network. Similarly, Yang et al. [21] uses a pre-trained graph neural network to learn object relationships. Liu et al. [12] used a neural-symbolic approach to conduct object navigation with a planner for high level reasoning and neural networks for object recognition. In contrast with these works, we are the first to focus on object navigation with multiple visual modalities.

8 CONCLUSION AND FUTURE WORKS

In this work, we implemented and compared embodied deep RL agents based on egocentric and allocentric visual input for the task of object navigation. We also proposed a dual-input CNN architecture that has the ability to capture both state-space perspectives



(a) Easy-Setting



(b) Hard-Setting

Figure 3: Episode -vs- Average Rewards

via a combined latent. We discovered that egocentric visual inputs are better for obstacle avoidance, thus lead to better early policy learning. Allocentric agents can learn a better policy in the long run due to the presence of complete state information. Our results also demonstrate that our dual CNN architecture has the ability to capture the most relevant information from each perspective for task completion. The network captures the benefit of both perspectives, ultimately resulting in greater rewards and shorter policy convergence. We have verified our hypothesis that embodied deep RL agents can be trained to capture information from multiple perspectives in a beneficial way. For future work, we would like to extend our approach to continuous actions with stochastic control. The added complexity yields a more realistic setting to account for factors such as fault tolerance. In addition, we would also like to use other types of multi-input architectures for data fusion. We want to incorporate more sensory information such as distance and sound into our embodied agent design.

REFERENCES

[1] Ki Uhn Ahn and Cheol Soo Park. 2020. Application of deep Q-networks for model-free optimal control balancing between different HVAC systems. *Science and Technology for the Built Environment* 26, 1 (2020), 61–74.

[2] Ramón Barber, Jonathan Crespo, Clara Gómez, Alejandra C Hernámdez, and Marina Galli. 2018. Mobile robot navigation in indoor environments: Geometric, topological, and semantic navigation. In *Applications of Mobile Robots*. IntechOpen.

[3] Shehroze Bhatti, Alban Desmaison, Ondrej Miksik, Nantas Nardelli, and Philip N. Siddharth. 2016. Playing Doom with SLAM-Augmented Deep Reinforcement Learning.. In *CoRP*.

[4] Jun Chen, Tingzhu Bai, Xiangsheng Huang, Xian Guo, Jianing Yang, and Yuxing Yao. 2017. Double-task deep q-learning with multiple views. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 1050–1058.

[5] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. 2021. A Survey of Embodied AI: From Simulators to Research Tasks. *CoRR* abs/2103.04918 (2021). arXiv:2103.04918 https://arxiv.org/abs/2103.04918

[6] Ziming Gao, Yuan Gao, Yi Hu, Zhengyong Jiang, and Jionglong Su. 2020. Application of deep q-network in portfolio management. In *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. IEEE, 268–275.

[7] Yuki Katsumata, Akira Taniguchi, Yoshinobu Hagiwara, and Tadahiro Taniguchi. 2019. Semantic Mapping Based on Spatial Concepts for Grounding Words Related to Places in Daily Environments. In *Frontiers in Robotics and AI*.

[8] Roberta L Klatzky. 1998. Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnections. In *Spatial cognition*. Springer, 1–17.

[9] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474* (2017).

[10] Minne Li, Lisheng Wu, Haitham Bou Ammar, and Jun Wang. 2019. Multi-view reinforcement learning. *arXiv preprint arXiv:1910.08285* (2019).

[11] Ruishan Liu and James Zou. 2017. The Effects of Memory Replay in Reinforcement Learning. *CoRR* abs/1710.06574 (2017). arXiv:1710.06574 http://arxiv.org/abs/1710.06574

[12] Xiaotian Liu and Christian Muise. 2021. A Neural-Symbolic Approach for Object Navigation. In *CVPR Embodied-AI Workshop*.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[14] Emilio Parisotto and Ruslan Salakhutdinov. 2017. Neural Map: Structured Memory for Deep Reinforcement Learning. In *arXiv:1702.08360*.

[15] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[16] Aviv Tamar, YI WU, Garrett Thomas, Sergey Levine, and Pieter Abbeel. 2017. Value Iteration Networks. In *NeurIPS*.

[17] Yuandou Wang, Hang Liu, Wanbo Zheng, Yunni Xia, Yawen Li, Peng Chen, Kunyin Guo, and Hong Xie. 2019. Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. *IEEE Access* 7 (2019), 39974–39982.

[18] Zhensong Wei, Yu Jiang, Xishun Liao, Xuewei Qi, Ziran Wang, Guoyuan Wu, Peng Hao, and Matthew Barth. 2020. End-to-End Vision-Based Adaptive Cruise Control (ACC) Using Deep Reinforcement Learning. *arXiv preprint arXiv:2001.09181* (2020).

[19] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. 2020. DD-PPO: LEARNING NEAR-PERFECT POINTGOAL NAVIGATORS FROM 2.5 BILLION FRAMES. In *ICLR*.

[20] Qiaoyun Wu, Dinesh Manocha, and Kai Xu Jun Wang. 2020. NeoNav: Improving the Generalization of Visual Navigation via Generating Next Expected Observations. In *AAAi*.

[21] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. 2019. Visual semantic navigation using scene priors. In *ICLR*.

[22] Eiling Yee, Michael N. Jones, and Ken McRae. 2018. Semantic Memory. In *Psychology Publications*.

[23] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3357–3364.