Reinforcement Learning and Place Cell Replay in Spatial Navigation

Chance J. Hamilton
Computer Science and Eng Dept
University of South Florida
Tampa, FL, USA
chamilton4@usf.edu

Pablo Scleidorovich
Computer Science and Eng Dept
University of South Florida
Tampa, FL, USA
pablos@usf.edu

Alfredo Weitzenfeld
Computer Science and Eng Dept
University of South Florida
Tampa, FL, USA
aweitzenfeld@usf.edu

***Abstract*—In the last decade, studies have demonstrated that hippocampal place cells influence rats' navigational learning ability. Moreover, researchers have observed that place cell sequences associated with routes leading to a reward are reactivated during rest periods. This phenomenon is known as Hippocampal Replay, which is thought to aid navigational learning and memory consolidation. These findings in neuroscience have inspired new robot navigation models that emulate the learning process of mammals. This study presents a novel model that encodes path information using place cell connections formed during online navigation. Our model employs these connections to generate sequences of state-action pairs to train our actor-critic reinforcement learning model offline. Our results indicate that our method can accelerate the learning process of solving an open-world navigational task. Specifically, we demonstrate that our approach can learn optimal paths through open-field mazes with obstacles.**

***Index Terms*—Place Cells, Hippocampal Replay, Reinforcement Learning, Actor-Critic, Latent Learning**

I. Introduction

A. Reinforcement Learning and Traditional Replay

Reinforcement learning (RL) is a machine learning approach that leverages past experiences to learn a policy to maximize a reward. RL has been successfully applied to robotic navigational tasks [1]–[4]. However, the number of trials required to learn a navigation task limits its applicability in real-world applications.

One way researchers have attempted to reduce the learning time of RL algorithms is to use experience replay. Experience replay is an offline-learning technique that stores previously experienced state-action sequences in a replay buffer and uses them to train the model. In 2017, Bruce *et al.* developed a model-free Deep-Q RL algorithm that uses experience replay to speed up learning [5]. Later in 2020, Jiang *et al.* showed that using experience replay, a Deep Q-learning algorithm can significantly reduce the number of navigational trails needed to learn optimal paths in unknown environments [6].

The traditional replay approach's limitation is the replay buffer's size. To benefit from experience replay, the model must store many previous experiences to see a decrease in the model's learning time. Furthermore, replay buffer models can only train on the same recorded events. No new experiences can be simulated using this method of replay.

*NSF IIS Robust Intelligence research collaboration grant #1703340

B. Hippocampal Place Cells

Research has shown hippocampal place cells play a vital role in spatial cognition and navigation [7], [8]. It is theorized that the activity of place cells encodes physical locations and implements neural mechanisms that support navigation and localization. Recently, place cells have provided a valuable framework for robotic navigational models [9]–[12]. Scleidorovich *et al.* [13] published a detailed analysis of how place cell size and place cell placement correlate to the learning rate of an Actor-Critic RL model. They showed that an appropriate mix of small and large place fields could optimize the path learned. However, their model did not utilize replay to speed up learning, thus requiring thousands of training episodes to learn an optimal path in an open maze.

C. Hippocampal Replay

Rodent experiments have shown that place cell sequences reactivate during periods of inactivity in the same or reverse order as experienced during a task. This phenomenon is known as hippocampal replay. [14], [15]. In 1994, Wilson and McNaughton conducted a study involving the reactivation of place cells in the rodents' hippocampus during sleep cycles is the same sequence of place cells that activated while the rodent occupied a specific spatial location. They theorized that the information experienced during an awake period is re-expressed in the hippocampus during a sleep cycle, which modifies the behaviors exhibited when the rodent is awake [16]. In 1996, Skaggs *et al.* showed that the reactivation of place cells during periods of rest act as a mechanism for long-term learning of spatial information [17].

D. Reinforcement Learning and Hippocampal Replay

Reinforcement learning algorithms require experiencing sequences of state-action pairs to learn. Mechanisms that synthesize these sequences have been exploited to speed up learning [18]. In 2005, Johnson and Redish developed such a mechanism. Specifically, a sparse matrix that stores connections between place cells. In theory, the stronger the relationship between two place cells, the more agent traversed the path between those place cells. Since the matrix encodes information about the paths traveled by an agent, we refer to it as a path matrix. Johnson and Redish showed their

model could find paths through a simple T-maze environment. However, their model was limited to solving the simple T-maze problem, not the more realistic open-world environment with obstacles. Additionally, they failed to show how their model would adapt to an environment with obstacles [19].

In 2020, Alabi *et al.* developed a one-shot RL Model that utilized replay to solve the Morris water maze task [11]. In their work, Alabi *et al.* uses reward cells to back-propagate replay from the latest activated place cell along its strongest synapses. Their model was able to solve the Morris Task in a one-shot manner. However, their model was unable to generalize to open environments with obstacles.

In this paper, we propose an extension of Johnson and Redish's replay model by enabling the path matrix to account for obstacles by inhibiting connections between place cells with an obstacle between them. We also present a novel stochastic method for generating state-action pairs that will be used for offline learning. Our model can find optimal paths for the Morris maze after one naive episode [20], even from unseen starting locations. We will further show our model's ability to adapt to open-field mazes with obstacles. The rest of this paper will proceed as follows; In Section II, we present our place cell replay model. In Section III, we outline the experiments we performed to evaluate our model. In Section IV, we summarize our model's results.

II. Place Cell Replay Model

This section will present our actor-critic RL algorithm with place cell replay. In Section II-A, we provide an overview of our computational representation of place cells and how they are utilized to represent the state space. Section II-B discusses our implementation of an Actor-Critic RL model and how it is used for the navigational task. Section II-C overviews our path matrix and its role in offline learning. Figure 1 shows the Actor-Critic RL with Place Cell Replay model architecture.

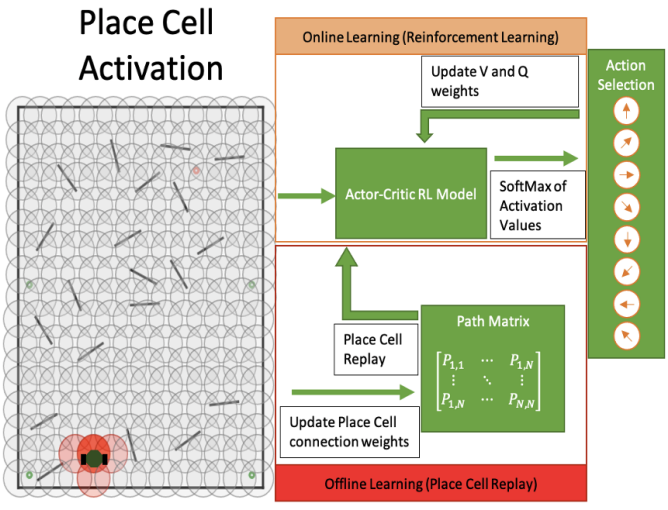


Fig. 1: Illustration of the architecture of our Actor-Critic RL with Place Cell Replay model.

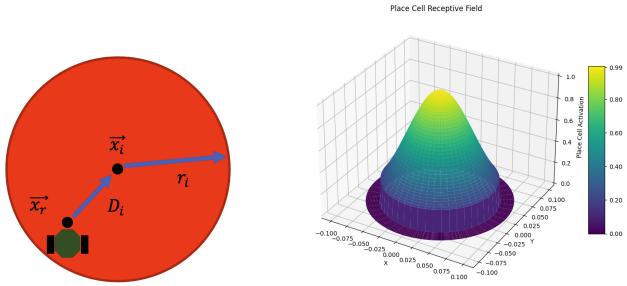
A. Place Cells

We represent our state space using place cells. We model our place cells using a normalized Gaussian radial basis function defined by their center $\vec{x_i}$ and their radius r_i [4], [21]. We assume the activation outside of a place cell's receptive field is zero and inside the radius is derived by a Gaussian kernel normalized by the activation of all place cells. The activation of a place cell is formally defined by Equations 1 and 2. Figure 2 shows a three-dimensional plot of a place cell's field and how the activation is calculated from the robot's position and place cell's center. For our experiments, we use a uniform distribution of 560 place cells with a radius of 16 cm that ensures the entire map is covered by place cell fields.

$$P'_{it} = \begin{cases} 0 & d_{it} < r_i \\ e^{-\frac{d_{it}^2}{r_i^2} ln(\alpha)} & \text{otherwise} \end{cases} \tag{1}$$

$$P_{it} = \frac{P'_{it}}{\sum_i P'_{it}} \tag{2}$$

Where P'_{it} refers to the calculated activation of place cell i at time t, while P_{it} is the normalized activation of place cell i at time t. The Euclidean distance between the center of place cell i (denoted as $\vec{x}_i$) and the robot's position at time t (denoted as $\vec{x}_r$) is represented as $d_{it} = ||\vec{x}_r - \vec{x}_i||$. We use r_i to denote the radius of place cell i. The constant α is utilized to specify the activation at the boundary of a place cell field, which happens when $d_{it} = r_i$. In this work, we set α to 0.001.



(a) (b)

Fig. 2: 2a Illustration of the place cell field based on the position of the robot $\vec{x_r}$, its distance D_i from the center of place cell i $\vec{x_i}$ with a radius of activation r_i.2b Plot of place cell activation field relative to the center of place cell i.

B. Actor-Critic Model

We extend the actor-critic model implemented by Sclei-dorovich *et al.* in 2020 [22]. Our model is trained in two modalities online and offline learning. The online learning algorithm takes the robot's position and returns an action to perform. Details regarding action selection and online learning can be found in Sections II-B2 and II-B1, respectively.

1) Online Reinforcement Learning: Our model utilizes an Actor-Critic Reinforcement Learning algorithm coupled with linear function approximations to simulate learning [4], [21]. During online learning, our model takes the robot's position and outputs one of eight allocentric actions denoted as a_t were $a_t \in A = \{0, ..., 7\}$. The action $j \in A$ represents the robot moving one step (8 cm) in the direction $\theta_j = \frac{\pi}{4} j$.

Given the robot's position, we calculate all place cell's activation using Equations 1 and 2. The activation of all place cells is then used to calculate the value for the current state and the preference for each action in the given state. Which is modeled as the linear function approximators V_t and Q_{jt}, respectively. The linear approximators V_t and Q_{jt} are defined by Equations 3 and 4, respectively.

$$V_t = \sum_i P_{it} V_{it} \tag{3}$$

$$Q_{jt} = \sum_i P_{it} Q_{ijt} \tag{4}$$

We represent the state value at time t as V_t, and V_{it} as the state value corresponding to place cell i at time t. Similarly, Q_{jt} represents the preference for action j at time t, and Q_{ijt} is the preference corresponding to place cell i for action j at time t.

2) Action Selection: The expected output of our model is one of eight allocentric actions, which is acquired by sampling the policy learned by the Actor-Critic model. Our model derives the policy for each action from the preferences obtained from Equation 4. We derive the policy for each action by applying a modified softmax function such that any action obstructed by obstacles has a zero probability, see Equation 5. The zero probabilities for actions resulting in obstacle collisions are to avoid damage to a physical robot. Our model performs action selection by randomly sampling the probability distribution produced by Equation 5.

$$\pi_{jt} = \frac{b_{jt} e^{Q_{jt}}}{\sum_i b_{it} e^{Q_{it}}} \tag{5}$$

We denote the probability of selecting action j at time t based on the actor's policy as π_{jt}. The Boolean variable b_{jt} accounts for actions that result in obstacle collision. In particular, if the action j at time t would lead to a collision, b_{jt} is set to 1. Otherwise, it is set to 0.

3) Bootstrap Error and Learning Rules: Our model uses a semi-gradient descent with a one-step bootstrap error as the learning rule for both the actor and critic [21]. We define the one-step bootstrap in Equation 6, the reinforcement error in Equation 7, and the rules for updating the actor and critic in Equations 8 and 9 respectively.

$$V'_t = \begin{cases} r_t + \gamma \sum_i P_{it} V_{i,t-1} & \text{if not at terminal state} \\ r_t & \text{otherwise} \end{cases} \tag{6}$$

$$\delta_t = V'_t - V_{t-1} \tag{7}$$

$$V_{it} = V_{i,t-1} + \alpha^V \delta_t \tag{8}$$

$$Q_{ijt} = V_{ij,t-1} + \alpha^Q \delta_t \tag{9}$$

Where V'_t represents the one-step return bootstrap. We use r_t to indicate the reward obtained at time t, which will be 1 if the agent reaches the goal's location and 0 otherwise. The discount factor, γ, is fixed at 0.95. The value associated with place cell i, calculated at time $t-1$, is denoted by $V_{i,t-1}$. The reinforcement learning error at time t is given by δ_t, and V_{t-1} represents the state value computed at time $t-1$. For our purpose, the learning rates for both α^V and α^Q are set to 0.4.

C. Place Cell Replay

To simulate state-action pairs to perform offline learning, we extend the path matrix presented by Johnson and Redish by introducing an inhibitor coefficient to prevent forming place cell connections across obstacles. We outline the details of our path matrix in Section II-C2. We also present a novel method for generating state-action pairs by creating a probabilistic distribution from the place cell connections, see Section II-C3. This differs from the work presented by Scleidorovich *et al.* in 2020, where only maximum place cell connections were used for state-action pair generation [22].

1) Path Matrix: During an online navigation episode, our model constructs a path matrix as a sparse matrix of size $N \times N$ where $N = 560$ is the number of place cells used to cover the environment. Each element in the path matrix, denoted as c_{ij}, represents the connection strength between place cell i to place cell j. After each time step, the connections between place cells are updated using Equation 10. We would like to point out that due to the relatively small size of the place cells (compared to the overall maze size) and the finite distance traveled during a single action, Equation 10 results in a single place cell forming only a few connections to other place cells that are relatively close to the start and endpoints of an action. For example, let us consider when the agent starts in a location at the bottom right of the maze and takes a single action forwards. Then only place cells located in the bottom right become active, and the remaining place cell activations are zero. Thus, no connections form between these nonactivated cells, resulting in a sparse path matrix. Although we derive equation 10 from Johnson and Redish's equations, our model averages the place cell's activation at times t and $t+1$, rather than just the activation at time $t+1$. In short, the more frequently the path traversed passes through place cell i to place cell j, the stronger the connection c_{ij} becomes. Thus, the trajectory frequently traversed by the robot will be encoded in the path matrix. Furthermore, the original equation generalizes to environments with obstacles intersecting place cell fields. Thus the inhibitor coefficient O_{ij}.

$$c_{ij}^{t+1} = O_{ij} \left(c_{ij}^t + tan^{-1} \left(\frac{P'_{i,t+1} + P'_{i,t}}{2} \cdot (P'_{j,t+1} - P'_{j,t}) \right) \right) \tag{10}$$

The equation above involves three variables. The variable c_{ij} denotes the connection strength from place cell i to place cell

j. The variable O_{ij} is a Boolean value inhibiting connections between place cells i and j if an obstacle exists between their centers. In other words, if the connection between place cells i and j intersects with an obstacle, O_{ij} is set to 0; otherwise, it is set to 1. Finally, P'_{it} represents the activity of place cell i at time t, as described in Equation 1.

2) Replay Events and Sequences: Our model uses the path matrix between each online episode to generate state-action sequences for performing offline learning. To better understand how we simulate state-action sequences, we will first describe a single replay event that produces a single state-action pair. Then we will discuss how multiple replay events are strung together to form state-action sequences.

To define a replay event, let us first consider the place cell i; we can construct a set of place cells C_i such that $C_i = \{j : c_{ij} | j > T \text{ and } i \neq j\}$. Where T is a place cell connection threshold set to 0.0001. Specifically, the set C_i represents all place cells (excluding place cell i) connected to place cell i greater than the threshold T. Next, we apply a softmax function to all connections c_{ij} found in set C_i. This produces a probability distribution for all positively connected place cells; see Equation 11. We randomly sample the distribution of place cell connections to find the next place cell. By randomly sampling the distribution, we can inject stochasticity into the generated state-action pairs, which ensures different state-action pairs are generated each time place cell i is considered. We parameterize the number of replay events that can be performed between each online episode, which we refer to as the replay budget.

$$\pi_{ij} = \frac{e^{c_{ij}}}{\sum_{C_i} e^{c_{ij}}} \tag{11}$$

The variables used in the above equation are as follows: π_{ij} denotes the probability of selecting the connection between place cells i and j, c_{ij} represents the connection weight from place cell i to place cell j, and C_i is the set of all place cells j where the connection weight is greater than or equal to 0.0001, and $i \neq j$.

Each replay event considers the starting place cell PC_t and produces the place cell PC_{t+1} with centers $\vec{p_t}$ and $\vec{p}_{t+1}$ respectively. We define the state as $\vec{p_t}$. To generate the action, we must select one of the eight allocentric actions which closely matches the transition from $\vec{p_t}$ to $\vec{p}_{t+1}$. Actions are derived from the angle between $\vec{p_t}$ and $\vec{p}_{t+1}$ and selecting the action closest to the angle. Formally, Equations 12 and 13 defines how offline action selection is performed.

$$\theta_t = \cos^{-1}\left(\frac{\vec{p}_{t+1} \cdot \vec{p_t}}{|\vec{p}_{t+1}||\vec{p_t}|}\right) \tag{12}$$

$$a_t = \left(\left\lfloor \frac{\theta_t}{\frac{\pi}{4}} \right\rceil \mod 8\right) \tag{13}$$

The variable p_t represents the center of the current place cell, while p_{t+1} represents the center of the next place cell. The variable θ_t is the angle between the two place cells, and a_t denotes the action closely aligned with the vector from p_t to p_{t+1}.

Now that we have discussed how a single replay event generates a state-action pair, we can discuss how we form state-action sequences for offline learning. To create a sequence of state-action pairs, we randomly select a place cell to start from, then perform a replay event to generate the first state-action pair and the next place cell. The next place cell produced by the replay event is used for the next replay event. This continues until one of the following conditions is met.

- The current place cell field contains the goal location.
- The next place cell to be considered already exists in the state-action sequence (i.e., a loop is formed).
- The replay sequence exceeds 1000 replay events.

Algorithm 1 outlines the method for generating the state-action sequences. Figure 3 illustrates the process for state-action sequence generation.

Algorithm 1 State-Action Generation

```
Randomly select PC_t
State_Action = [ ]    ▷ Array to store state-action pairs
while Not in a terminal state do
    PC_{t+1}, a_t = Replay Event from PC_t
    append (p_t, a_t) to State_Action
    PC_t = PC_{t+1}
end while
```

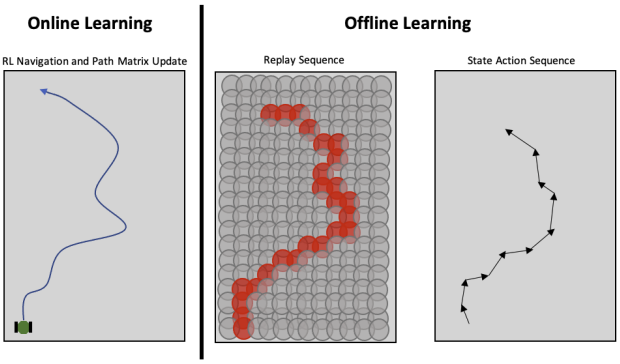


Fig. 3: Offline replay learning. The robot traverses a path during an online episode using the RL algorithm and updates the path matrix. After the online episode, the path matrix simulates replay events converted into a state action pair sequence for training the RL algorithm.

3) Offline Reinforcement Learning: Each state-action sequence simulated is used to train the actor-critic RL model using the same semi-gradient decent with a one-step bootstrap error leaning rule outlined in Section II-B3.

III. Experiments

To evaluate the performance of the model, we devised two experiments. Section III-A presents the evaluation metric used to gauge the effectiveness of place cell replay for offline learning. In Section III-B, we construct an experiment to highlight

the capabilities of the path matrix. Our next experiment is designed to show how our model can rapidly learn optimal paths in an open environment, even from unseen starting locations; see Section III-C.

A. Evaluation Metric

To evaluate the model, we consider how optimal is the path learned by the model. We quantify a path's optimality using the *extra step ratio* (also referred to as path indirectness) [12], [13], [23]. This metric takes the number of additional steps needed to reach the goal from a starting location and normalizes it with the number of steps required by the optimal path. Equation 14 formally defines the *extra step ratio*. We normalize the number of extra steps to ensure they are independent of the shortest path's length. We can therefore think of the *extra step ratio* as the number of extra steps taken per required step.

Consequently, the ideal *extra step ratio* is 0, which implies that the robot learned a path with the same number of steps as the shortest path. However, we consider any *extra step ratio*$<$ 1 optimal for our purposes. That is to say, the robot only performed at most one extra step for every required step. We would like to point out that the extra step ratios should always be greater or equal to 0. However, a negative *extra step ratio* may be produced due to our approximation of the shortest path. We discretize the space into a 1 mm square grid and derive the shortest path using the A* algorithm [24]. This may produce an *extra step ratio* less than 0 implying that model found a better solution than the A* algorithm.

$$e_T = \frac{A_T - M}{M} \tag{14}$$

The variable e_T corresponds to the extra step ratio for a specific robot in trial T, whereas A_T indicates the total number of actions the robot takes in the same trial. Finally, M represents the minimum number of actions required to reach the goal in the respective maze.

B. Properties of the Path Matrix

As stated in Section II-C2, by averaging the place cell's activation at time t and $t+1$ we ensure that the path matrix strengthens connections between place cells the more frequently a trajectory traverses through the respective fields. Furthermore, a desirable property emerges due to the $P'_{j,t+1} - P'_{j,t}$ term in Equation 10. Specifically, suppose a trajectory contains detours, such as loops and deviations from the optimal path. The place cell connections tend to maintain strong connections along the optimal path.

To demonstrate this property, we program the robot to follow two paths, each with a detour somewhere along the trajectory. Despite the detours, both paths ensure the robot traverses the optimal path at least once. Using the methods outlined in Section II-C2, we create a path matrix that encodes the experiences of both paths. We then plot the path matrix by showing each place cell's strongest connection. Figure 4 illustrates this property while considering two types of detours.

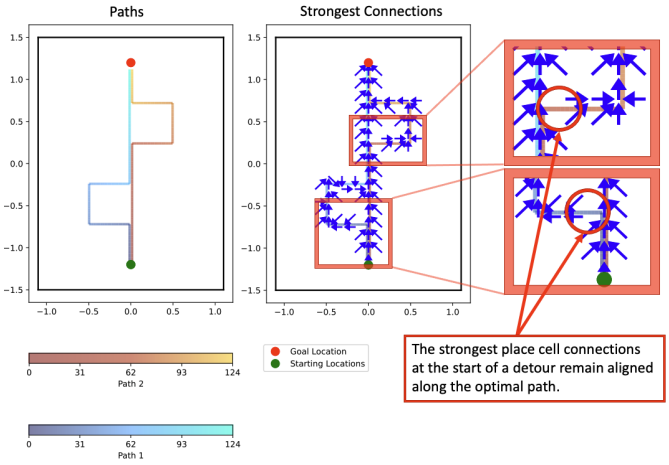


Fig. 4: Figure on the left shows the two paths traversed by the robot; each path contains a detour. The middle image shows the strongest place cell connections recorded in the path matrix. The figures on the right illustrate how the place cell connections at the beginning of the detours still remain aligned along the optimal path.

By observing the strongest connections shown on the right, we can see that the place cells closest to the beginning of the detour maintain the strongest connection along the optimal path despite the detours.

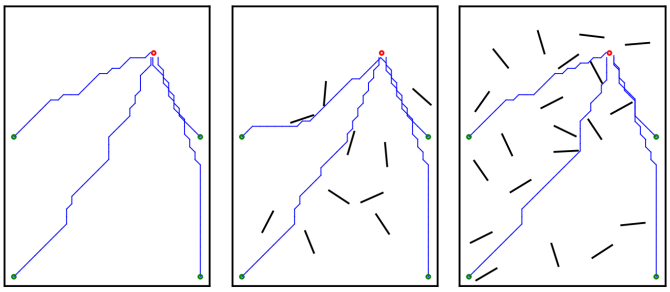
C. Rapid Learning of Optimal Paths

Out next set of experiments showcase the rapid learning of optimal paths, even from unseen starting locations. For this experiment, we consider three open maze configurations, one with no obstacles (similar to the Morris water maze [20]), one with ten obstacles randomly placed in the environment, and another with 20 obstacles. Each maze consists of four starting locations and one static goal location. Figure 5 shows the aforementioned maze configurations.

For each maze, we allow the robot to run four online episodes, each time starting from a new starting location. The first online episode is considered naive, as the robot has no prior knowledge of the maze. During the naive episode, a path matrix is created. A set amount of replay events are simulated between the subsequent online episodes. The generated state-action sequences are used to train our actor-critic model. We compare how the model performs when no replay is performed and when 16 million replay events are simulated between the episodes. We repeat the experiment 50 times for each maze and each replay budget allotment.

IV. Results

We provide quantitative and qualitative analysis for the experiment outlined in Section III-B. In Section IV-A we compare the performance of our model when place cell replay is utilized vs when it is not. In Section IV-B, we analyze the quality of the paths learned during each online episode.



(a) Maze: M000 **(b)** Maze: M104 **(c)** Maze: M204

Fig. 5: **(a)**, **(b)**, and **(c)** shows illustrations of the mazes with 0 obstacles, ten obstacles, and 20 obstacles, respectively. We represent the four starting locations as green dots and the static goal location as red dots. The blue trajectories show the shortest paths obtained using an A* algorithm.

A. *Quantitative Analysis*

To gauge the effectiveness of place cell replay we consider two allotments for the replay budget for each of our open-world mazes: no replay events and 16 million. We ran each replay budget and maze configuration 50 times in simulation and recorded the *extra step ratio* when the robot started from each of the starting locations. We display the recorded *extra step ratio* from each starting location as a box plot and observe the improvement after the first episode when place cell replay is performed. Figure 6 shows a box plot of the extra step ratios distribution for all 50 simulated experiments during each episode. The left column shows the model's performance when no replay was performed after the first episode for each of the three mazes we considered. Similarly, the right column shows the model's performance with a 16 million replay events replay budget. We would like to mention that we experimented with various amounts for the replay budget and found that a replay budget of 16 million ensures that the model learns optimal paths even after the first episode.

When observing the plots 6a and 6b, we see that place cell replay produces a drastic improvement in the *extra step ratio*. In both 6a and 6b, we see that episode 0 performs similarly. However, after place cell replay is used for offline learning (i.e., after the first episode), a significantly more optimal path is traversed despite starting from a new location. Furthermore, plot 6b shows that for episodes 1, 2, and 3 (which start from different locations), the model produces optimal paths when place cell replay is performed. Similar observations are seen when comparing the performance of mazes M104 and M204. We note that in the more complicated mazes M104 and M204 (mazes with obstacles), there is slightly more variance in the *extra step ratio* of the paths recorded for episode 3. We attribute this to the relative distance from the starting location to the goal location. The episode corresponds to the starting location in the upper right of each maze, which has the shortest optimal path derived from the A* algorithm. When the path matrix is first constructed, the region containing the shortest path from the last starting location is typically unexplored. Thus, when place cell replay is performed, the expected reward

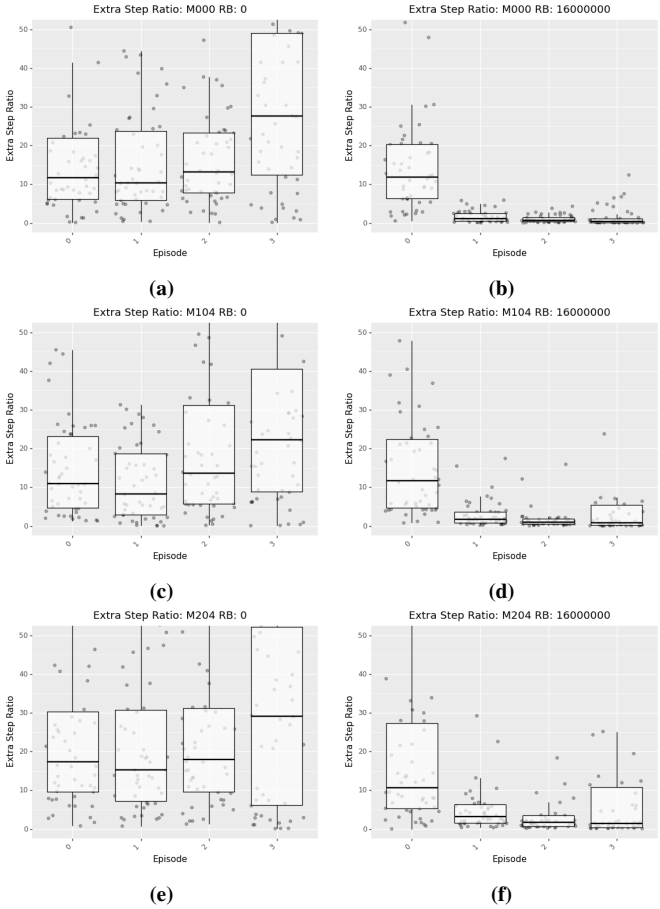


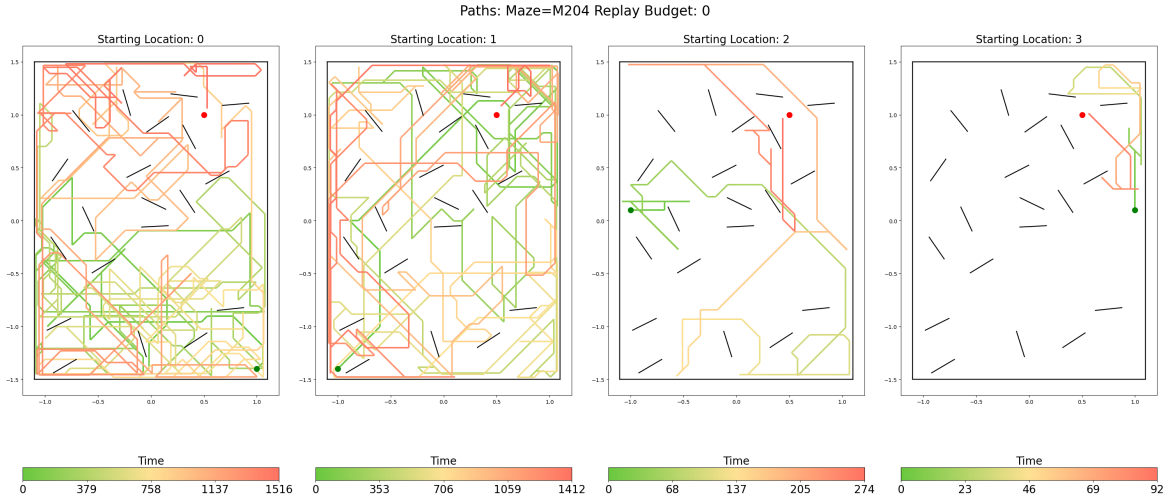
Fig. 6: The plots in **(a)**, **(c)**, and **(e)** show the models performance (in terms of the extra step ratio) when the replay budget is 0 for mazes M000, M104, and M204, respectively. Similarly, **(b)**, **(d)**, and **(f)** show the model's performance when the replay budget is 16 million.

in those states is significantly smaller than in other states (towards the other starting locations). This leads the model to believe that states further left are more rewarding than up and to the left, which causes the robot to move further left than required. An example of this is shown in Section IV-B.

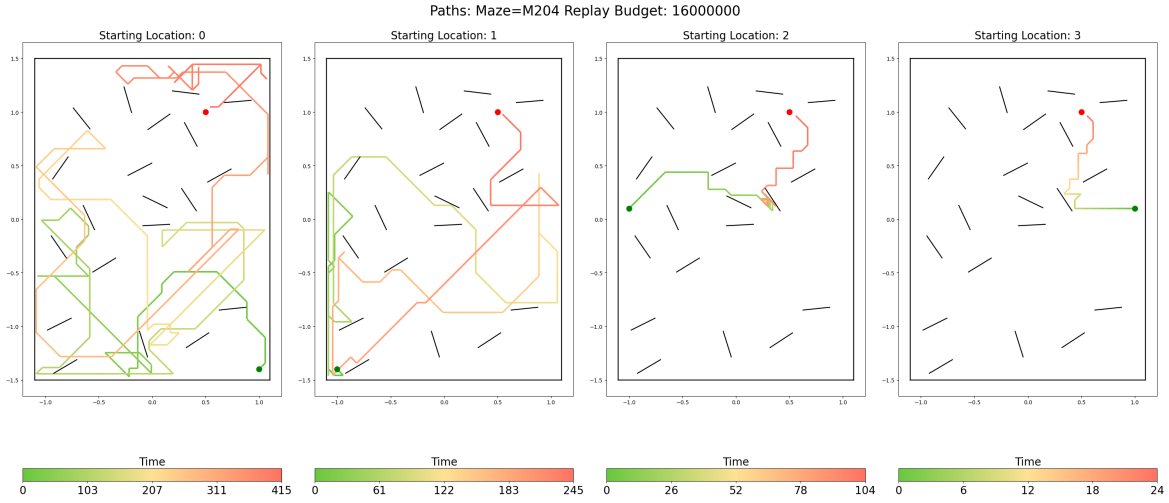
Table I reports the median extra step ratios for each starting position and maze. Our results show that the median extra step ratio obtained when the model performs place cell replay always obtains a lower medial extra step ratio when compared to the model when no place cell replay is utilized. Furthermore, for mazes M000 and M104, the model using place cell replay generally finds optimal paths by the third starting location.

B. *Qualitative Analysis*

To visualize the effects of place cell replay on the paths produced by our model, we plot the paths from each of the starting locations that the robot traverses in the most complicated maze, M204. Figure 7 illustrates the path traversed by an agent during a single trial. Figure 7a traces the four paths traversed during each episode of the trial when no place cell replay was performed. Similarly, Figure 7b traces the paths traversed during each episode when place cell



(a)



(b)

Fig. 7: (a) Displays the paths performed by the robot when no place cell replay is performed in maze M204. Similarly, **(b)** shows the paths with place cell replay is performed.

Median Extra Step Ratios		
Maze and Starting Locations	*RB = 0*	*RB = 16e6*
M000: 1	11.750	11.765
M000: 2	10.338	1.054
M000: 3	13.174	0.609
M000: 4	27.577	0.308
M104: 1	10.906	11.656
M104: 2	8.297	1.698
M104: 3	13.609	0.913
M104: 4	22.308	0.885
M204: 1	17.406	10.719
M204: 2	15.189	3.257
M204: 3	17.957	1.652
M204: 4	29.154	1.461

TABLE I: Summery of median extra step ratios for each starting point in each maze when the replay buffer is 0 and 16 million.

replay was performed. We can see that the paths exhibit a significant tendency toward exploration when no place cell replay is performed. This is intuitively understandable, as the model has minimal training experience and still needs more exploration to learn a path to the goal. On the other hand, when place cell replay is performed, we see a significant decrease in exploration and a considerable increase in navigation by the second episode. Again, this is understandable since the model is exposed to several magnitudes of more simulated experiences. As mentioned in Section IV-A, Figure 7 shows how the previous experiences force the trajectory towards the left of the maze. Specifically, when the robot departs from starting location 3, it heads towards the left, as most of the past experiences happened on the left side of the maze.

V. Discussion

We have developed a novel approach to performing replay for an Actor-Critic RL algorithm. Our model uses place cells

to represent the state space and uses navigational experiences to form and strengthen place cell connections. The place cell connections are stored in a path matrix that encodes path information experienced during online navigation. Unlike previous works, our path matrix can account for environmental obstacles using the obstacle inhibitor coefficient presented in Equation 10. Furthermore, we have presented a novel algorithm for generating state-action sequences from the path matrix that ensures more of the state space is explored during offline learning. We have shown that using place cell replay can rapidly and significantly improve the optimality of paths learned by our actor-critic RL model. The place cell replay proposed in work directly applies to robot navigation, enabling robots to rapidly learn paths through an unknown cluttered environment.

In the future, we plan to transfer our model to a physical robot and investigate if place cell replay can provide a viable mechanism for rapid exploration and navigation of unknown environments. In future work, we plan to include a visual SLAM algorithm to activate place cells as the robot navigates a new environment.

ACKNOWLEDGMENT

This work was funded by NSF IIS Robust Intelligence research collaboration Grant No. #1703225 at the University of South Florida and Grant No. #1703440 at the University of Arizona, entitled "Experimental and Robotics Investigations of Multi-Scale Spatial Memory Consolidation in Complex Environments."

REFERENCES

[1] X. Ruan, D. Ren, X. Zhu, and J. Huang, "Mobile robot navigation based on deep reinforcement learning," in *2019 Chinese control and decision conference (CCDC)*. IEEE, 2019, pp. 6174–6178.

[2] N. ALTUNTAŞ, E. Imal, N. Emanet, and C. N. Öztürk, "Reinforcement learning-based mobile robot navigation," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 24, no. 3, pp. 1747–1767, 2016.

[3] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[4] V. R. Konda and J. N. Tsitsiklis, "[NIPS 2000] Actor-critic algorithms," *Advances in Neural Information Processing Systems*, 2000.

[5] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "One-shot reinforcement learning for robot navigation with interactive replay," *arXiv preprint arXiv:1711.10137*, 2017.

[6] L. Jiang, H. Huang, and Z. Ding, "Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 1179–1189, 2019.

[7] J. O'Keefe and J. Dostrovsky, "The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat." *Brain research*, 1971.

[8] R. G. Morris, P. Garrud, J. a. Rawlins, and J. O'Keefe, "Place navigation impaired in rats with hippocampal lesions," *Nature*, vol. 297, no. 5868, pp. 681–683, 1982.

[9] M. L. Alonso, *Multi-Scale Spatial Cognition Models and Bio-Inspired Robot Navigation*. University of South Florida, 2017.

[10] M. Llofriu, G. Tejera, M. Contreras, T. Pelc, J. M. Fellous, and A. Weitzenfeld, "Goal-oriented robot navigation learning using a multi-scale space representation," *Neural Networks*, vol. 72, pp. 62–74, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2015.09.006

[11] A. Alabi, A. A. Minai, and D. Vanderelst, "One Shot Spatial Learning through Replay in a Hippocampus-Inspired Reinforcement Learning Model," pp. 1–8, 2020.

[12] A. Alabi, D. Vanderelst, and A. Minai, "Rapid Learning of Spatial Representations for Goal-Directed Navigation Based on a Novel Model of Hippocampal Place Fields," pp. 1–27, 2022. [Online]. Available: http://arxiv.org/abs/2206.02249

[13] P. Scleidorovich, M. Llofriu, J. M. Fellous, and A. Weitzenfeld, "A computational model for spatial cognition combining dorsal and ventral hippocampal place field maps: multiscale navigation," *Biological Cybernetics*, vol. 114, no. 2, pp. 187–207, apr 2020.

[14] C. Pavlides and J. Winson, "Influences of hippocampal place cell firing in the awake state on the activity of these cells during subsequent sleep episodes," *Journal of neuroscience*, vol. 9, no. 8, pp. 2907–2918, 1989.

[15] G. R. Sutherland and B. McNaughton, "Memory trace reactivation in hippocampal and neocortical neuronal ensembles," *Current opinion in neurobiology*, vol. 10, no. 2, pp. 180–186, 2000.

[16] M. A. Wilson and B. L. McNaughton, "Reactivation of hippocampal ensemble memories during sleep," *Science*, vol. 265, no. 5172, pp. 676–679, 1994.

[17] W. E. Skaggs, B. L. McNaughton, M. A. Wilson, and C. A. Barnes, "Theta phase precession in hippocampal neuronal populations and the compression of temporal sequences," *Hippocampus*, vol. 6, no. 2, pp. 149–172, 1996.

[18] B. Peng, X. Li, J. Gao, J. Liu, K.-F. Wong, and S.-Y. Su, "Deep dyna-q: Integrating planning for task-completion dialogue policy learning," *arXiv preprint arXiv:1801.06176*, 2018.

[19] A. Johnson and A. D. Redish, "Hippocampal replay contributes to within session learning in a temporal difference reinforcement learning model," *Neural Networks*, vol. 18, no. 9, pp. 1163–1171, 2005.

[20] R. G. Morris, "Spatial localization does not require the presence of local cues," *Learning and motivation*, vol. 12, no. 2, pp. 239–260, 1981.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[22] P. Scleidorovich, M. Llofriu, J. M. Fellous, and A. Weitzenfeld, "A Computational Model for Latent Learning based on Hippocampal Replay," *Proceedings of the International Joint Conference on Neural Networks*, 2020.

[23] P. Scleidorovich, J.-M. Fellous, and A. Weitzenfeld, "Adapting hippocampus multi-scale place field distributions in cluttered environments optimizes spatial navigation and learning," *Frontiers in Computational Neuroscience*, vol. 16, pp. 1 039 822–1 039 822, 2022.

[24] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968.