SNAP:Successor Entropy based Incremental Subgoal Discovery for Adaptive Navigation

Rohit K. Dubey
Technical University of Munich
Germany
rohit.dubey@tum.de

Samuel S. Sohn
Rutgers University
USA
samuel.sohn@rutgers.edu

Jimmy Abualdenien
Technical University of Munich
Germany
jimmy.abualdenien@tum.de

Tyler Thrash
Saint Louis University
USA
tyler.thrash@slu.edu

Christoph Hölscher
ETH Zurich
Switzerland
choelsch@ethz.ch

André Borrmann
Technical University of Munich
Germany
andre.borrmann@tum.de

Mubbasir Kapadia
Rutgers University
USA
mubbasir.kapadia@rutgers.edu

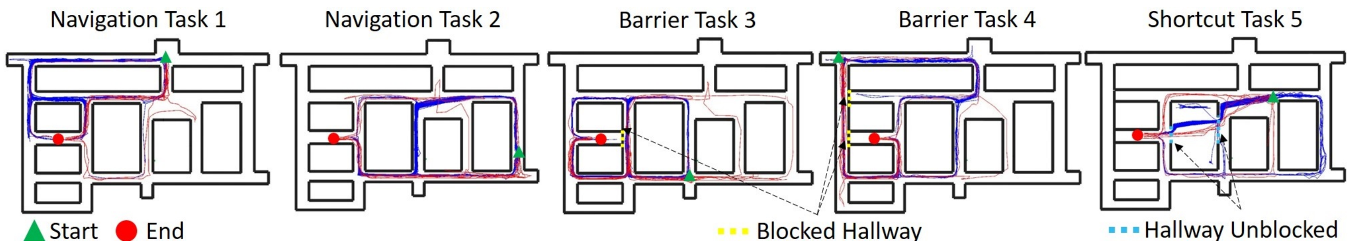

Figure 1: Visualization of trajectories generated by optimal agent and human participants for five wayfinding tasks. Agent's trajectories are visualized in blue and human participants trajectories in red.

ABSTRACT

Reinforcement learning (RL) has demonstrated great success in solving navigation tasks but often fails when learning complex environmental structures. One open challenge is to incorporate low-level generalizable skills with human-like adaptive path-planning in an RL framework. Motivated by neural findings in animal navigation, we propose a **S**uccessor e**N**tropy-based **A**daptive **P**ath-planning (**SNAP**) that combines a low-level goal-conditioned policy with the flexibility of a classical high-level planner. **SNAP** decomposes distant goal-reaching tasks into multiple nearby goal-reaching sub-tasks using a topological graph. To construct this graph, we propose an incremental subgoal discovery method that leverages the highest-entropy states in the learned Successor Representation. The Successor Representation encodes the likelihood of being in a future state given the current state and capture the relational structure of states based on a policy. Our main contributions lie in discovering subgoal states that efficiently abstract the state-space and proposing a low-level goal-conditioned controller for local navigation. Since the basic low-level skill is learned independent of state representation, our model easily generalizes to novel environments without intensive relearning. We provide empirical evidence that the proposed method enables agents to perform long-horizon sparse reward tasks quickly, take detours during barrier tasks, and exploit shortcuts that did not exist during training. Our experiments further show that the proposed method outperforms the existing goal-conditioned RL algorithms in successfully reaching distant-goal tasks and policy learning. To evaluate human-like adaptive path-planning, we also compare our optimal agent with human data and found that, on average, the agent was able to find a shorter path than the human participants.

CCS CONCEPTS

• **Computing methodologies → Planning with abstraction and generalization**.

KEYWORDS

Goal-conditioned RL, Robot navigation, Option discovery, Adaptive path-planning, Hippocampus

1 INTRODUCTION

Humans and other animals can quickly navigate unfamiliar and complex environments while adapting to environmental changes. This process of navigation can be conceptually divided into locomotion (i.e., low-level control) and wayfinding (i.e., high-level planning). Locomotion and wayfinding are driven by a neural circuit in human and animal brains that facilitates vector-based navigation and flexible path-planning. Vector-based navigation [Bush et al. 2015; Edvardsen 2015] is the low-level strategy by which a navigator observes their final goal and moves directly towards it in the absence of obstacles. In the presence of obstacles, humans and other animals have been found to switch to an adaptive path-planning strategy by which the navigator moves between subgoals using a topological graph employing vector navigation between subgoals [Bush et al. 2014; Edvardsen et al. 2020].

To mimic these adaptive navigation behaviors with intelligent agents, classical path-planning algorithms such as A*, Dijkstra's, D*, and its variant requires prior information of the environment in the form of a dense navigational graph. A more extensive and complicated environment will result in higher complexity, memory utilization, and computation time. Moreover, the above-mentioned foundational path-planning algorithms consistently produce the shortest path, which animals or humans do not observe in an under-explored environment. Recent works in RL attempts to model adaptive path-planning by employing both goal-conditioned planning [Eysenbach et al. 2019; Huang et al. 2019; Pong et al. 2018; Schaul et al. 2015; Zhang et al. 2020] and hierarchical RL algorithms [Kulkarni et al. 2016; Nachum et al. 2018; Ramesh et al. 2019]. A goal-conditioned RL agent learns policies to reach a final goal state by subdividing the distant navigation task into smaller sub-tasks towards different subgoals. These subgoals are typically elected from a topological graph using a shortest path heuristic but can be based on other common heuristics (e.g., the path with least turning). To improve efficiency, hierarchical RL abstracts the path-planning task into multiple levels. This abstraction is especially useful for planning in complex environments or with sparse rewards. While both goal-conditioned and hierarchical RL models learn a local policy to reach nearby states, they often fail to navigate successfully during barrier and shortcut tasks when environment structure changes.

For long-distance goals, the likelihood of transitioning to any other future state from the current state can be formulated using Successor Representations (SR) [Dayan 1993]. SR encodes the relational structure of states based on a chosen policy. Previous research has demonstrated that SR is one way in which action states are encoded by the human brain [Russek et al. 2017; Stachenfeld et al. 2017]. Recently, in [Ramesh et al. 2019], SR has also been shown to abstract the environment by identifying subgoals that are well-distributed throughout the environment and unique. However, SR is unable to re-evaluate changes to the state space that may result from environmental changes such as paths being blocked by obstacles [Gershman 2018; Momennejad et al. 2017].

To overcome these challenges, we propose **S**uccessor e**N**tropy-based **A**daptive **P**ath-planning (**SNAP**) that combines a low-level goal-conditioned policy with the flexibility of a classical high-level planner that decomposes distant goal-reaching tasks into multiple nearby goal-reaching sub-tasks. Similar to [Eysenbach et al. 2019; Huang et al. 2019], our method combines classical path-planning and deep RL with the aim of tackling long-distance, sparse reward navigation tasks. We train the low-level goal-conditioned policy using a direction vector computed with nested grid cells. In parallel with learning this low-level policy, the agent also learns a high-level planner that leverages a topological graph. This topological graph abstracts the environment state space by the incremental discovery of subgoals that gradually identify states representing bottleneck locations (i.e., door/opening in a corridor). Over several iterations, the agent learns the SR of the visited states and distinct clusters are learned from these states. Subgoals are chosen as the states with the highest entropy within each identified cluster. We also propose an Approximate Reward Representation (ARR) that is learned in parallel with SR and encodes the state-to-state transition reward matrix. Specifically, ARR maintains the minimum distance between pairs of states using a replay buffer that helps transform the sparse reward tasks to dense-reward tasks and that formulates the edges of the topological graph in terms of estimated distances between subgoals. Our approach makes the following contributions:

- We formulate a novel Approximate Reward Representation in conjunction with a new interpretation of SR for identifying subgoals that aid in efficient learning for a low-level goal-conditioned RL controller.
- We propose a human-like path-planner that switches between two modes of navigation depending on obstructions in the environment.
- In a behavioral study, we provide empirical evidence of the similarity between the subgoals of the new SR method and the subgoals chosen by human participants on a variety of fundamental navigation tasks.

Our method **SNAP** supports adaptive and global navigation in complicated environments. We showcase via multiple simulations and compare with human data that an efficient selection of state representations for iterative subgoal discovery can finesse the expensive sequential state-state navigation, identify novel shortcuts, and adapt behavior during goal/reward revaluations.

2 RELATED WORK

This section begins by briefly highlighting the evidence found in neuroscience that supports human navigation and developing a mental representation of the environment. Then we discuss relevant RL methods proposed earlier in the literature to solve long-horizon navigational tasks similar to our proposed approach.

2.1 Neural Basis of Navigation

Recent studies in neuroscience have provided compelling evidence for the theoretical notion of a cognitive map and its neural instantiation in the hippocampus and Medial Entorhinal Cortex [Breathnach 1980; McNaughton et al. 2006]. Multiple neuron population types such as place cells [O'Keefe and Dostrovsky 1971], grid cells [Hafting et al. 2005], head-direction cells [Taube et al. 1990], and border/boundary-vector cells (BVC) [Barry et al. 2006; Lever et al. 2009; Solstad et al. 2008] have been discovered and considered to collectively form a dynamic neural circuit for animals that aid in self-localization, dynamic path-planning, and vector-based navigation (see Section 1 of the Supplementary Material for more details). In 2005, Grid cells were identified that fire periodically at regular intervals as an animal moves around in a space [Hafting et al. 2005]. Such periodic firing is thought to provide a multi-scale representation of the local environment, which assists in the localization of its position [Fiete et al. 2008; Hafting et al. 2005] and in vector-based navigation [Fiete et al. 2008]. Successor Representation (SR) exhibits the properties of place and grid cells observed in rodents and can perform reward revaluation as observed in humans. However, SR is insensitive to changes in the transition structure (i.e., transition revaluation). [Gershman 2018; Momennejad et al. 2017]. Boundary Vector Cells (BVC), also known as boundary cells or border cells, are neurons found in the hippocampus that activates in the presence of an environment boundary or obstacle at a particular distance or direction from the animal [Barry et al. 2006; Lever et al. 2009]. BVC can signal the presence of a nearby barrier and thus facilitate path adjustments before actual contact with the obstacle. In this paper, we consider the role of replays during path-planning when BVCs are activated in proximity to an environment boundaries or the agent's sensory input signifies a change in the environment such as the addition of a barrier.

2.2 Options Discovery in Reinforcement Learning

The options framework proposed by Sutton and colleagues [Sutton et al. 1999] models temporally extended actions for mitigating the difficulty of solving difficult Markov-Decision Processes (MDPs) in RL. The careful design of options is crucial to exploiting the benefits of the options framework and can be used to speed-up the policy learning process. The discovery of options largely revolves around identifying either a bottleneck state [McGovern and Barto 2001; Menache et al. 2002; Şimşek and Barto 2004; Stolle and Precup 2002] or a landmark state [Harutyunyan et al. 2019; Machado et al. 2017a,b; Ramesh et al. 2019]. The key difference between bottleneck and landmark states is that the former represents states corresponding to bottlenecks such as doors or small gaps that have a high visitation count and high centrality. In contrast, landmark states represent regions that are well-connected and evenly distributed. Within built environments, bottleneck and landmark states are duals of each other, i.e. two bottleneck states (e.g., doors) are connected by a landmark state (e.g., a room) and two landmark states are connected by a bottleneck state. Our approach is closer to bottleneck states because human paths are often near-optimal [Buecher et al. 2009] and shortest-paths are guaranteed to pass through bottleneck states, but not landmark states.

2.3 Hierarchical & Planning based Navigation

Similar to options frameworks, hierarchical RL (HRL) techniques are motivated by learning a set of primitive skills to solve complex tasks. Most HRL approaches combine a low-level controller with a high-level planner. The task of the high-level planner is to learn a global layout of the environment and produce a sequence of landmarks or subgoals for the low-level controller to reach one at a time [Eysenbach et al. 2019; Kulkarni et al. 2016; Nasiriany et al. 2019]. For example, SoRB [Eysenbach et al. 2019]) employs HRL to bridge the gap between planning and deep RL to solve long horizon tasks. SoRB decomposes the task of reaching a distant goal into a sequence of easier tasks by employing graph search with a goal-conditioned policy. Similarly, in [Xu et al. 2021], the authors propose to use an approximate 2D map of novel environments during navigation that does not require any learning. They train a deep generative model to generate intermediate subgoals in the observation space that low-level goal-conditioned RL can then use to achieve distant navigation tasks. Our approach is similar to the HRL approach because it uses graph-based path-planning as proposed in SoRB [Eysenbach et al. 2019] and MAP-UGR [Huang et al. 2019]. However, our approach differs from these two methods in two significant ways. First, our approach aims to produce a human-like path-planner and thus only utilizes a small number of subgoals instead of a large number of subgoals compared to other works. Second, unlike SoRB, wherein the main insight is to use RL in constructing the graph, our approach learns a vector navigation skill to reach each subgoal using RL. Moreover, our approach does not depend on learning a reliable distance function (i.e., estimating the distance to a goal) and is thus less prone to local navigation errors. Finally, we extend the long horizon sparse reward navigation task to accommodate dynamic changes in an environment, which was not addressed in the above approaches.

2.4 Goal-conditioned Reinforcement Learning

Several variants of goal-conditioned RL methods have been proposed in the literature [Andrychowicz et al. 2017; Eysenbach et al. 2019; Huang et al. 2019; Kaelbling 1993; Mirowski et al. 2016; Pong et al. 2018; Schaul et al. 2015]. For example, Hindsight Experience Replay (HER) [Andrychowicz et al. 2017] combines off-policy RL with goal-relabeling to enhance the sample complexity and robustness of goal-conditioned policies. HER was improved by explicitly implementing a connection between model-based, and model-free learning [Pong et al. 2018] and by randomly sampling goals from the learned latent space to use as replay goals for off-policy Q-learning rather than limiting only to states observed along the sampled trajectory, facilitating efficient learning [Nair et al. 2018]. Another popular goal-conditioned model-free RL algorithm is Universal Value Function Approximators (UVFA). While HER and UVFA both use a single neural network to represent value and successfully solve short-horizon tasks, they both fail to model long-range tasks. To tackle such problems, algorithms such as [Huang et al. 2019; Mirowski et al. 2016; Zhu et al. 2017] have been proposed to combine a locally robust controller with a high-level planner.

Our proposed framework **SNAP** follows a similar approach as UVFA [Andrychowicz et al. 2017], and SoRB [Eysenbach et al. 2019] by building a low-level controller using goal-conditioned RL. Our method differs from the above methods in the following ways. (1) The goal-conditioned policy in our method does not depend on distance estimates and has improved sample efficiency over UVFA. (2) It does not depend on an approximate external 2D map. (3) It learns the policy simultaneously while discovering subgoals and thus reduces training time.

3 APPROACH

In principle, model-free goal-conditioned RL agents can solve long-horizon tasks, but in practice, they struggle to reach goals that are distant [Nasiriany et al. 2019]. To overcome this challenge, we combine goal-conditioned policies (i.e., a low-level controller) with a high-level planner that decomposes a distant goal-reaching task into k shorter subtasks to subgoals. The overview of the proposed method **SNAP** is summarized in Figure 2 and Algorithm 1. Our approach consists of four main components.

- **Nested Grid Cells for Egocentric Direction** (i.e., computation of allocentric vector direction between agent and the goal. Section 3.2).
- **Low-level Controller** (i.e., learning goal-conditioned policy to execute vector-based navigation between two nearby subgoals. Section 3.3).

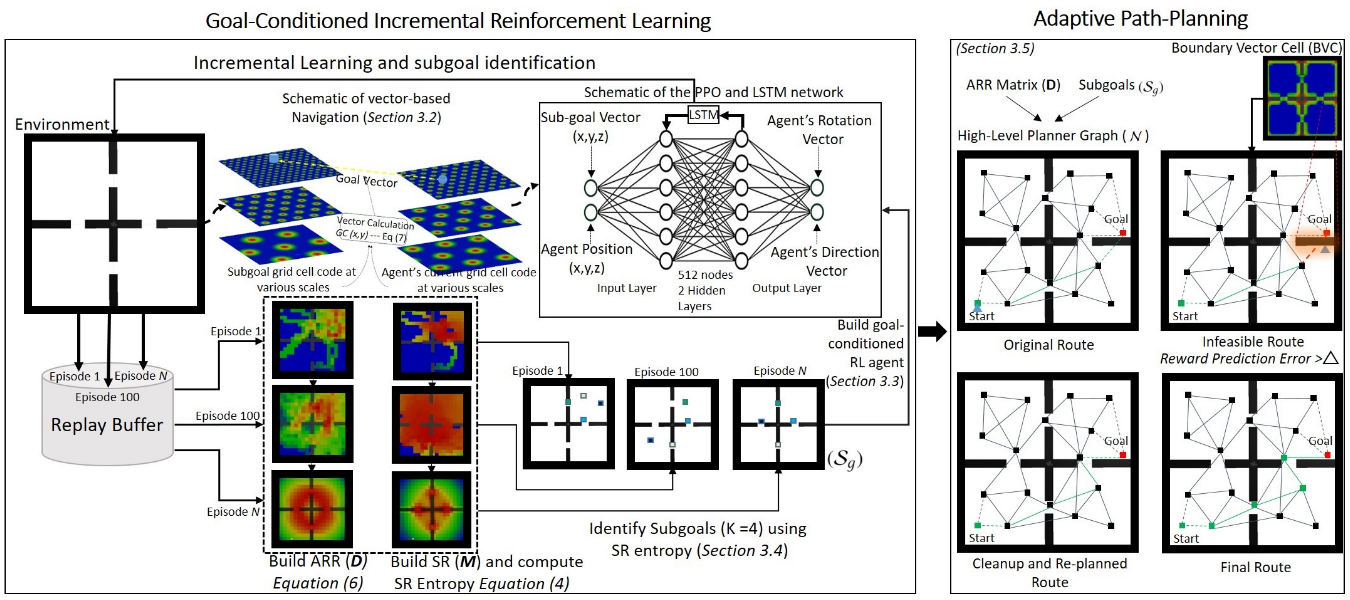
Goal-Conditioned Incremental Reinforcement Learning
Incremental Learning and subgoal identification
Environment
Schematic of vector-based Navigation (Section 3.2)
Goal Vector
Vector Calculation GC (x,y) --- Eq (7)
Subgoal grid cell code at various scales
Agent's current grid cell code at various scales
Schematic of the PPO and LSTM network
LSTM
Sub-goal Vector (x,y,z)
Agent's Rotation Vector
Agent Position (x,y,z)
Agent's Direction Vector
512 nodes 2 Hidden Layers
Input Layer
Output Layer
Build goal-conditioned RL agent (Section 3.3)
Episode 1
Episode N
Episode 100
Replay Buffer
Episode 1
Episode 100
Episode N
Build ARR (D) Equation (6)
Build SR (M) and compute SR Entropy Equation (4)
Episode 1
Episode 100
Episode N
$(\mathcal{S}_g)$
Identify Subgoals (K =4) using SR entropy (Section 3.4)
Adaptive Path-Planning
(Section 3.5)
Boundary Vector Cell (BVC)
ARR Matrix (D)
Subgoals $(\mathcal{S}_g)$
High-Level Planner Graph ($\mathcal{N}$)
Goal
Start
Original Route
Infeasible Route
Reward Prediction Error >△
Cleanup and Re-planned Route
Final Route

Figure 2: Overview of steps involved in SNAP.

- **Incremental Subgoal Discovery** (i.e., discovery of landmark states that guides humans and animals path towards a goal using high-level planner. Section 3.4).
- **Adaptive path-planner** (i.e., high-level planner that can produce human-like optimal and flexible navigation behavior to discover detours and novel shortcuts. Section 3.5).

3.1 Preliminaries

The overview of the proposed method **SNAP** is presented in Algorithm 1. To model a low-level controller, we consider a goal-reaching agent interacting with an environment. The environment can be formulated as a Markov Decision Process (MDP) described by the tuple $(S, A, \mathbf{T}, \mathcal{R}, \gamma)$ where S is a set of agent-environment states, A is a set of available actions to the agent, $\mathbf{T}$ is a state transition function, $\mathcal{R}$ is a reward function $\mathcal{R}(s'|s,a)$ that indicates the reward in transitioning to state s' from state s by taking action a, and $\gamma \in [0, 1]$ is a discount factor that reduces the weight of rewards obtained further in the future (lines 1 - 5). The agent's state space is a discretized 2D grid of the entire space comprising of 1×1 cells. At each time step t, the agent performs an action guided by its policy and receives an intrinsic reward (lines 6 - 11). The agent moves at a constant speed $|v|$ in any direction. The set of available actions, $A = [|v|0^\circ, |v|360^\circ]$. The agent receives a reward of -1/MAX_{steps} for taking each step and a reward of +1 for reaching a subgoal or goal state. We convert the sparse-reward problem into a dense reward by leveraging ARR. The agent receives an intrinsic reward along with a negative reward of -0.01 if it enters a state which is close to obstacles (i.e., computed using BVC input) (lines 12 - 19).

Learning a goal-conditioned policy is a key component of our method. In this work, we employ a policy gradient-based RL method. Instead of learning Q-Values or Value functions such as in standard RL methods, policy gradient methods learn a policy function directly. Our implementation uses Proximal Policy Optimization (PPO), which performs similar to or better than state-of-the-art approaches and is easy to tune [Schulman et al. 2017]. We refer readers to [Schulman et al. 2017] for PPO's implementation details. The sections below describe the formulation of the approximate reward representation that maintains the minimum distance between pairs of states using a replay buffer and successor representation that support the discovery of subgoals.

3.1.1 Approximate Reward Representation.

To compute the reward ($\mathcal{R}$), we propose to simulate how the human brain maps discounted rewards to states while learning an environment. We assume that an approximate reward for experienced trajectories between every pair of visited states in the environment is encoded. To maintain such mapping, we propose the Approximate Reward Representation (ARR). ARR ($\mathbf{D}$) encodes a pairwise distance between two locations during the agent's exploration. This distance is employed as an intrinsic dense reward for an agent to efficiently learn a goal-conditioned low-level controller. The agent receives a reward $r_{intrinsic} = (\mathbf{D}[s, s_{goal}] - \mathbf{D}[s', s_{goal}])$ where s is the agent's current state, s' is the agent's next state, and s_{goal} is the agent's target goal state. In our model, the ARR is a $n \times n$ matrix where n is the number of states. $\mathbf{D}(s, s')$ encodes the approximate best reward as a function of time-steps/distance from the goal (Equation 1), where a smaller distance yields an exponentially higher reward. λ is a constant (λ = 0.1) and $\mathcal{D}(s')$ is the number of time-steps/distance between a state s' and the goal/subgoal. We assume the ARR matrix $\mathbf{D}$ is initialized to the identity matrix $\mathbb{I}$, meaning $\mathbf{D}(s, s') = 1$ if $s = s'$, and $\mathbf{D}(s, s')$ = 0 if $s \neq s'$. This is because we assign the reward of transitioning from the same state to itself as 1.

$$\mathcal{R}(s') = e^{-\lambda * \mathcal{D}(s')} \tag{1}$$

$$\mathbf{D}'(s, s') = \begin{cases} \mathcal{R}(s'), & \text{if } \mathcal{R}(s') \geq \mathbf{D}(s, s') \\ \mathbf{D}(s, s'), & \text{otherwise} \end{cases} \tag{2}$$

Algorithm 1 Incremental Subgoal Discovery and Policy Learning

```
Inputs: $N$ (Number of Episodes)
Inputs: $k$ (Total number of Subgoals)
Inputs: $\gamma$ (Discount factor)
Inputs: $\Pi$ (Goal Conditioned RL Policy)
Inputs: $RB$ (Replay Buffer)
Inputs: $\epsilon$ (Max. steps per episode)
Inputs: $s_s, s_g$ (start state, goal state)
 1: $\mathbf{A} \leftarrow \{\}$ //Adjacency Matrix
 2: $\mathbf{T} \leftarrow \{\}$ //Transition Matrix
 3: $\mathbf{D} \leftarrow \{\}$ //Approximate Reward Representation
 4: $\mathbf{M} \leftarrow \{\}$ //Successor Representation Matrix
 5: $RB \leftarrow \{\}$ //Replay Buffer
 6: $StepCounter = 0$
 7: for $n \leftarrow 0$ to $N-1$ do
 8:     for $StepCounter \leftarrow 1$ to $\epsilon$ do
 9:         if $(n > 0)$ then
10:             if (Path($s_s, s_g$)) then
11:                 $RB \leftarrow$ Navigate($O, \Pi, \mathbf{D}, \mathcal{N}$)
12:             else
13:                 $RB \leftarrow$ RandomAction($\Pi$)
14:     for $i \in |RB|$ do
15:         for $j \in |RB|$ do
16:             if $RB(s_i \leftarrow s_j = 1)$ then
17:                 $\mathbf{A}(i,j) = 1$
18:     for $i, j$ to $|S|$ do
19:         $\mathbf{T}[s_i, s_j] \leftarrow \frac{\mathbf{A}[s_i,s_j]}{\sum_{j=0}^{|S|} \mathbf{D}[s_i,s_j]}$
20:     $\mathbf{M} \leftarrow (\mathbb{I} - \gamma \mathbf{T})^{-1}$
21:     $\mathbf{D} \leftarrow$ ComputeARR($RB, \mathbf{D}$)          Eq. (8)
22:     $\mathbf{SE} \leftarrow$ ComputeSR-Entropy($\mathbf{M}$)
23:     $O \leftarrow$ ClusterSR($\mathbf{M}, k, \mathbf{SE}$)
24:     $\mathcal{N} \leftarrow$ BuildGraph($O$)
25:     $RB \leftarrow \{\}$
26: $\mathbf{SE} \leftarrow$ ComputeSR-Entropy($\mathbf{M}$)
27: $O \leftarrow$ ClusterSR($\mathbf{M}, k, \mathbf{SE}$)
28: $\mathcal{N} \leftarrow$ BuildGraph($O$)
29: return $O, \mathcal{N}$
```

In our proposal, the SR and ARR Matrices are learned independently and in parallel.

3.1.2 Successor Representation. As proposed in [Stachenfeld et al. 2017], any value function can be represented as a linear combination of the reward function $\mathcal{R}$ and the SR $\mathbf{M}$ (Eq. 3).

$$V(s) = \sum_{s'} \mathbf{M}(s, s')\mathcal{R}(s') \tag{3}$$

$\mathbf{M}$ is initialized to the identity matrix I, meaning $\mathbf{M}(s, s') = 1$ if $s = s'$, and $\mathbf{M}(s, s') = 0$ if $s \neq s'$. When the transition probability matrix is known, we can compute the SR as a discounted sum over transition matrices raised to the exponent t (Eq. 4). The matrix T^t is the t-step transition matrix, where $T^t(s, s')$ is the probability of transitioning from s to s' in t steps.

$$\mathbf{M} = \sum_{t=0}^{\infty} \gamma^t T^t \tag{4}$$

This sum is a geometric matrix series, and for $\gamma < 1$, it converges to the following finite analytical solution:

$$\mathbf{M} = \sum_{t=0}^{\infty} \gamma^t T^t = (I - \gamma T)^{-1} \tag{5}$$

where I is the identity matrix. In all of our simulations, SR was computed analytically from the transition matrix using Equation 5 (lines 12 - 18 in Algorithm 1).

3.2 Nested Grid Cells for Egocentric Direction

Grid cells are believed to represent an animal's coordinates system in 2D and 3D space [Edvardsen 2018; Stella and Treves 2015]. The unique repetitive hexagonal pattern formed by grid cells can be used for geometric computations (i.e., direction and distance to a goal from the current location). In this paper, the grid cell was generated by employing the model suggested by [Solstad et al. 2006]. Equation 6 computes the hexagonal grid cell pattern at any given location (x, y) by intersecting three waves at $\pi/3°$ apart.

$$\begin{aligned} GC(x,y) = max\Bigg[0, -0.2 + \prod_{d=0}^{2}\Bigg(1 + cos\Bigg((x - o_x).\\ \frac{2\pi}{S}.cos(r + d.\frac{\pi}{3}).\frac{2}{\sqrt(3)} + (y - o_y).\\ \frac{2\pi}{S}.sin(r + d.\frac{\pi}{3}).\frac{2}{\sqrt(3)}\Bigg)\Bigg)\Bigg] \end{aligned} \tag{6}$$

Three parameters, s, r, and o, characterize grid cells here. s denotes the scale of the pattern (i.e., the distance between two hexagonal peaks), r is its orientation, and o is the 2D offset parameters from some reference point. Figure 3 visualizes a top-down view of a grid cell formation at various scales of 10,15,23, and 34 meters in a square box of 102 square meters. Orientation r is fixed to $\pi/2$ and a 2D offset o(x,y) is set to $(0, 0)$. Vector navigation using one grid-scale formulation poses multiple restrictions. If the grid-scale is small, for example, 10 meters, the direction to any location outside the range of this scale will be error-prone (Figure 1 (D) in [Edvardsen 2018]). At a larger scale, vector navigation will be jittery and less accurate. The input vector direction (θ) for an agent to a subgoal is calculated by measuring the vector between the two respective locations represented by two separate grid cell populations. One of the grid cell populations encodes the agent's current location, whereas the other population encodes the subgoal's location (Equation 7).

$$\theta = GC(x_{agent}, y_{agent}) - GC(x_{subgoal}, y_{subgoal}) \tag{7}$$

3.3 Low-level Controller using Goal-Conditioned Policy

Navigation between the distant start and goal states is facilitated by a low-level goal-conditioned controller and a high-level planner. The controller operates during each training episode, and the planner operates between episodes. We first describe the initial episode. For the low-level controller, the agent begins at the start state and randomly moves between states for $\epsilon = 3000$ steps while being

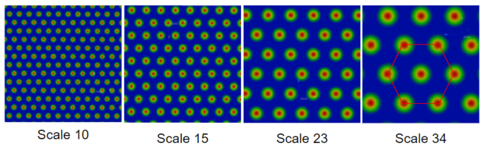

Figure 3: We employ nested grid cells that form a repetitive hexagonal pattern to compute vector direction between agent and a subgoal. Grid cells are generally observed to formulate at scales that are in a ratio of 1.5.

guided by its rewards (Section 3.1). During this movement, the agent maintains a replay buffer of its visited states. This replay buffer is used to compute the SR and ARR. At the end of each episode, the learned SR is used to discover k temporary subgoals (Section 3.4). The identified subgoals are treated as nodes, and a topological graph ($\mathcal{N}$) is formulated (Section 3.5). The agent's start position is kept constant at the bottom leftmost corner of an environment with a random state for a destination during learning. An optimal path between a given pair (o,d) is computed using a Dijkstra's Algorithm. If a path exists, the goal-conditioned agent is given a subgoal target closest to the agent in the identified path. At the start of training, the ARR matrix is under-developed, and hence most of the time, no edge exists between nodes and does not result in an optimal path between a pair (o,d). In such a scenario, the set of subgoals is sorted by the descending order of ARR sums (i.e., ARR Sum of subgoal s_i is $\sum_{j=0}^{|S|} \mathbf{D}[s_i, s_j]$). If no path exists, the goal-conditioned agent is given a target, the first subgoal in the ordered subgoal list. Once the agent reaches a subgoal, the target is moved to the next subgoal in the list (see Supplementary Video).

To learn a policy, we employ PPO proposed in [Schulman et al. 2017], but one could also use any other off-the-shelf algorithm. PPO uses a convolutional neural network (CNN) to approximate the ideal function that maps an agent's observations to the best action an agent can take in a given state (see Supplementary Material for hyperparameter details). In our approach, the agent receives two observations: the egocentric vector angle (θ) computed using Equation 7 towards the target subgoal and its current location. Aside from assisting in identifying subgoal states with high centrality for policy learning, another benefit of ARR is in affording a dense reward in the context of sparse reward navigational tasks. The reward function is formalized in Equation 8 (φ_s is the ARR of state s).

$$r_{\varphi_s}(s_t, a_t, s_{t+1}) = \varphi_s(s_{t+1}) - \varphi_s(s_t) \tag{8}$$

The agent is rewarded positively if it takes a step closer to the target subgoal. The agent is penalized for taking each step. The episode terminates after ϵ timesteps or if the agent successfully reaches the final goal, whichever occurs first. The proposed incremental goal-conditioned RL learning-based approach enables learning without extrinsic reward and explores distant states, which may not be feasible using only primitive actions.

3.4 Incremental Subgoal Discovery

During navigation, humans and animals discover and employ landmarks to guide their path towards a destination [Dubey et al. 2019; Epstein and Vass 2014]. In an unfamiliar environment, this process of landmark discovery is incremental and continuous. A landmark identified earlier with limited exploration can also be replaced if a more informative landmark is identified later for the same space. We propose an incremental discovery of subgoals (i.e., bottleneck states) to model landmark discovery heuristics similar to those observed in humans. A similar approach is proposed in [Ramesh et al. 2019]. An essential difference between our method and the approach taken in [Ramesh et al. 2019] is the definition of a subgoal. We consider landmarks to have high centrality in order to facilitate human-like path planning [McGovern and Barto 2001; Moradi et al. 2010], and accordingly, we ensure that the corresponding states are frequented by agents along successful paths. Identifying effective landmark states will enable faster exploration and faster convergence to the optimal policy. In Figure 5 we showcase that the proposed SR-Entropy based method outperforms SR-Options based method in coverage of state-space with respect to mean square error difference from optimal ARR.

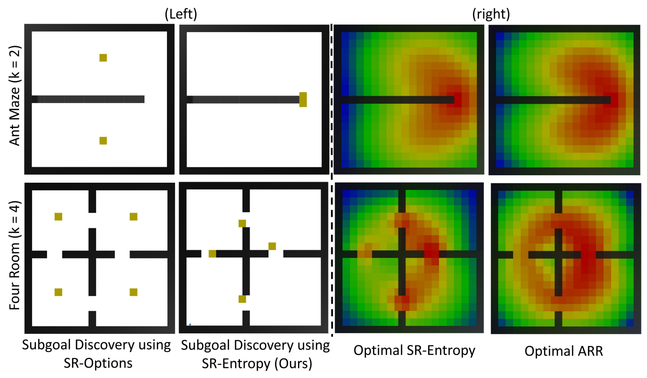
(Left)
(right)
Ant Maze (k = 2)
Four Room (k = 4)
Subgoal Discovery using SR-Options
Subgoal Discovery using SR-Entropy (Ours)
Optimal SR-Entropy
Optimal ARR

Figure 4: (left) Computation of subgoals using SR-Options vs. SR-Entropy (Ours) on two different environments. (right) Visualization of optimal SR-Entropy and Approximate Reward Representation of states.

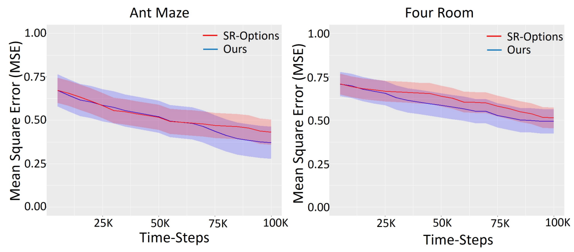

Figure 5: Mean Square Error (MSE) difference between incremental ARR and optimal ARR with subgoal-directed agent exploration using SR-Options and SR-Entropy-based subgoal identification on two different environments.

The first step required to discover subgoals is to learn SR. To learn the successor representation, we maintain a replay buffer of visited states per episode. The replay buffer is reset to an empty sequence at the beginning of each episode. The SR matrix ($\mathbf{M}$) is computed by updating adjacency ($\mathbf{A}$) and transition matrices ($\mathbf{T}$) based on the replay buffer using Equation 5. At the end of each episode, all visited states are clustered using the farthest point sampling algorithm (K-means++). K-means++ eliminates the shortcoming of K-means that it is dependent on the initialization of centroids [Arthur and Vassilvitskii 2006]. Clustering over SR produces clusters of states spread across the visited state-space, with each cluster comprising densely connected states. Ideally, identifying the cluster centers as subgoals will facilitate exploration of the region in a state-space with vastly different successor states. Employing them as nodes for path-planning will primarily produce longer paths that are generally not observed in human navigation. Instead, we compute k subgoals as states in each cluster that have maximum entropy. The high entropy states also resemble states with high betweenness centrality that is crucial to learn new skills in RL [Moradi et al. 2010]. In Figure 4, we present the subgoals using SR-Options (i.e., cluster's centroid) and SR-Entropy (i.e., our) approaches. The process of exploring states and clustering operates in an iterative fashion by which each episode results in visits to unexplored states followed by SR updates, SR clustering, and SR-Entropy computations, respectively. To perform K-means++ clustering, we need to provide the value for k. We consider k as a parameter that is specified manually beforehand. Ideally, with optimal state-space knowledge, k should directly correspond to the number of bottleneck states (i.e., doors and gaps in a cluttered environment) for every distinct cluster. However, in our simulations, we notice that having a larger value of k ($2 * k_{optimal}$) supports quicker policy learning. Moreover, due to the basic nature of the proposed RL agent that lacks human-like vision, larger values of k support vector navigation more efficiently.

3.5 Adaptive Navigation

During a goal-oriented navigation task in a cluttered environment, humans most often aim to reach a nearby landmark state to circumvent obstacles and to continue towards the goal [Shamash et al. 2020; Verma and Mettler 2017]. In the absence of any obstacles, humans identify their final goal and move directly towards it. Therefore, we propose a combined vector-based navigation and topological navigation strategy to incorporate similar switching behavior. In the absence of a nearby boundary or obstacle, the distinct process of goal-vector navigation guides the movement direction based on the learned goal-conditioned policy towards a goal. In the presence of a barrier or obstacle identified using BVCs, the navigation strategy switches to topological navigation, and a subgoal state is selected from the optimal path using Graph $\mathcal{N}$. The BVC model proposed by [Barry et al. 2006] is employed in our work. A detailed description of the computation of BVCs is provided in the Supplementary Materials (Section 1.3). Given a set of subgoals (S_g), the ARR matrix ($\mathbf{D}$), and start and goal positions (o, d), our next step is to formulate a topological graph ($\mathcal{N}$). ARR is used for estimating the distance between two states (i.e., $d(s_i, s_j) = \mathbf{D}[s_i, s_j]$). If $d(s_i, s_j) \le \tau$, an undirected weighted ($w = d(s_i, s_j)$) edge is connected between node s_i and s_j. The threshold τ ensures that subgoals are connected locally, such that the low-level controller is likely tractable. Otherwise, there is a risk that a obstacle exists between subgoals, with which the controller would likely struggle. We begin by finding the shortest path using Dijkstra's algorithm between o and d. This process outputs a list of subgoal nodes in $\mathcal{N}$. The first subgoal in this list is the target subgoal given to the low-level RL agent. If the subgoal is reached, a new high-level path is planned, and the process repeats until the agent reaches d. Sometimes, a path towards the next subgoal may be blocked, which is sent to the agent as a Reward Prediction Error (RPE) via BVCs. If the cumulative RPE reaches a certain predefined threshold, the edge towards the current subgoal is removed from $\mathcal{N}$, and a new path is computed. This allows an agent to circumvent getting stuck at obstacles and to avoid following the last edge for any future paths since the deleted edge no longer exists in $\mathcal{N}$.

4 EXPERIMENT & RESULTS

4.1 Environments, Baselines, and Evaluation Metrics

4.1.1 Environment. We compare our method with two approaches that tackle a goal-reaching task in sparse-reward settings on the three most commonly used environments (i.e., Ant Maze, Four Room, and DeepMind layouts) [Andrychowicz et al. 2017; Eysenbach et al. 2019; Huang et al. 2019].

4.1.2 Baseline 1. Hindsight Experience Replay (HER) [Andrychowicz et al. 2017] proposes a goal-relabeling that reuses previous experience to train a Universal Value Function Approximator (UVFA) in a sparse reward setting.

4.1.3 Baseline 2. MAPping state space using landmarks for Universal Goal Reaching (MAP-UGR) [Huang et al. 2019] proposes a sample-based method to dynamically map the visited state space and to solve long-range goal-reaching problems. Their approach to forming a map for path-planning using subgoals/landmarks is similar to our approach but differs in terms of training a low-level goal-conditioned RL policy.

4.1.4 Evaluation Metrics. *(1) Success Rate:* The ratio of successful goal-reaching attempts over total attempts: $\frac{n_{success}}{n_{total}}$. The total number of steps is counted by summing numbers of steps during training and testing. *(2) Cumulative Reward:* The average sum of all rewards over training. *(3) Episode length:* The length of the simulation at the end of which the agent reaches a terminal state.

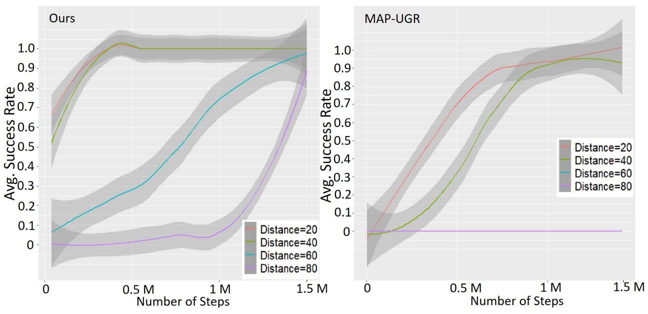

Figure 6: Simulating multiple goal-reaching tasks of increasing difficulty on a large (50 × 50) Ant maze environment.

4.2 Simulation Results

To evaluate the goal-reaching capability of all approaches, we sample random goal locations from a fixed start location situated at the

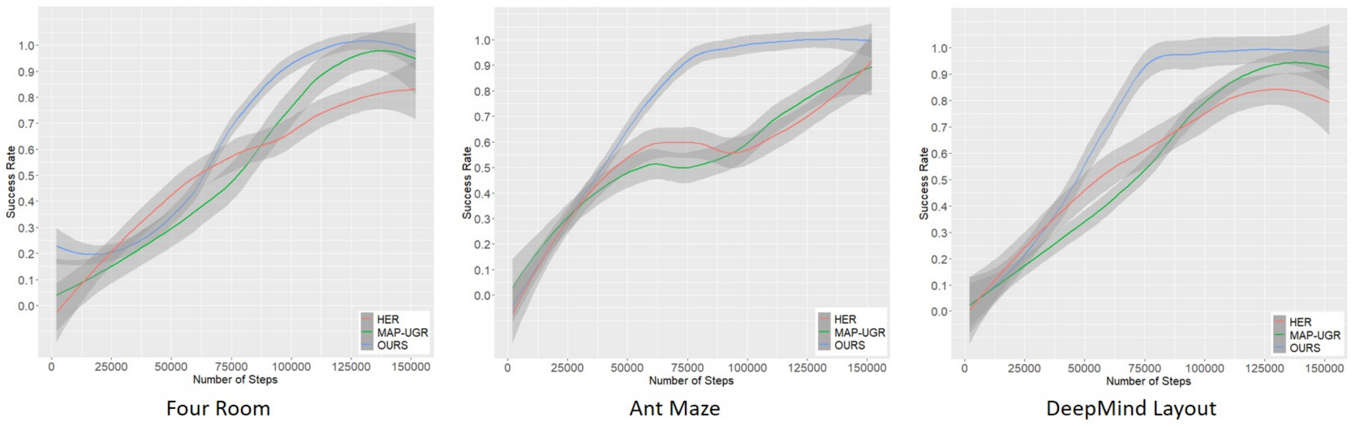

Figure 7: Simulation results showcasing average success rate on three different layouts using HER [Andrychowicz et al. 2017] and MAP-UGR [Huang et al. 2019] approaches and the proposed approach.

bottom left corner in each environment. Goal-conditioned RL is used to learn a policy by training an agent for 2000 steps and testing the learned policy for ten episodes. Each episode ends if an agent reaches the goal or runs out of maximum allowed timesteps (200 steps for HER and MAP-UGR and 3000 steps for our approach). An average success rate of ten tests is computed after every 2000 steps. In Figure 7, we showcase the average success rate (in %) for all three environments. The number of subgoals was predetermined based on a simulation conducted by varying the number of subgoals in each environment (see Figure 1 in Supplementary Material). The subgoal numbers for Four Room, Ant Maze, and DeepMind layouts were 10, 10, and 16. In all environments, our approach achieves a high success rate earlier in training compared to recent methods. Moreover, the policy learned in our approach is stable. Our approach produces a relatively consistent 100% success rate after learning the policy because it is not dependant on distance estimates between two nearby landmarks as in the other two approaches. We further test our method and the above two approaches in reaching distant goals of increasing difficulty in an Ant Maze environment. We manually assign a task's difficulty as the distance (in steps) between start and goal locations (20, 40, 60, 80) as shown in Figure 6. The agent start location is at the bottom-left of the layout, and goal states are chosen randomly for each distance. In Figure 6, We plot the average success rate to reach different difficulty levels of goals. Unlike the HER and MAP-UGR, we observe that our method takes significantly fewer steps to reach a high average success rate in all distant goal-reaching tasks. In addition, the other two approaches take a long time to reach a high success rate, and the number of steps taken is proportional to the goal difficulty level. Moreover, the success rate for these two approaches becomes unstable for distant goals (i.e., 60 and 80 meters), which is not observed in our method. In Table 1 we present the mean success rate for distant reaching goal in comparison to MAP-UGR. We can notice that the proposed method significantly outperforms MAP-UGR. Our method quickly learns a goal-conditioned policy and is not sensitive to the goal distance. Please refer to the supplementary video that showcases the important functionality of the proposed agent with an example of distant goal-reaching tasks in a complicated environment.

Table 1: Mean success rate for a distant goal-reaching tasks at a distance of 20,40,60, and 80 steps using a policy trained for 1 million timesteps using MAP-UGR [Huang et al. 2019] and our approach.

	20		**40**		**60**		**80**	
	Mean	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
Ours	1	0	1	0	0.835	0.163	0.307	0.305
MAP-UGR	0.985	0.034	0.907	0.138	0	0	0	0

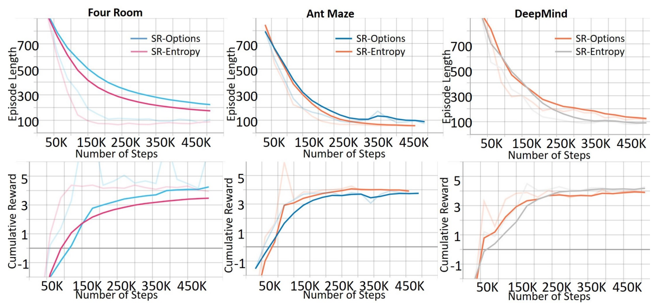

Figure 8: Impact of two different clustering approaches on overall cumulative reward and episode length during training.

4.3 Ablation Study

We investigate important factors that influence our proposed approach. (1) Subgoal sampling strategies. (2) Goal-conditioned policy with and without a high-level planner (3) Baseline (PPO) Vs. Baseline + Grid Cells Vs. Baseline + Grid Cells + BVCs. We report the finding for ablation study 3 in the Supplementary Material (see Figure 2 in Supplementary Section 2 respectively).

4.3.1 Subgoal Sampling Strategies. We employ two different subgoal sampling strategies. Specifically, we use the incremental SR-Options method proposed in [Ramesh et al. 2019] and the incremental SR-Entropy-based approach presented here. In Figure 8, we demonstrate the merit of both approaches over three layouts. For all three layouts, we notice a lower episode length for SR-Entropy-based incremental learning. We also see a higher cumulative reward for two of three layouts using SR-Entropy. One probable reason why SR-Entropy-based subgoals consistently resulted in lower episode lengths is the location of identified subgoals (i.e., mostly at the bottleneck states), which enabled an agent to reach a goal along a relatively shorter path than SR-options-based subgoals.

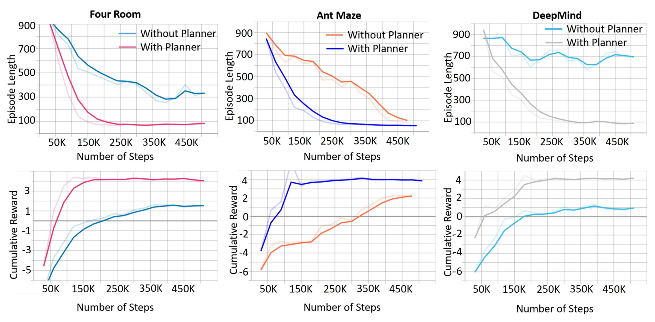

Figure 9: Impact of the graph-based planner with and without goal-conditioned RL agent on overall cumulative reward and episode length during training.

4.3.2 With and Without Planner. In Figure 9, we showcase the impact of using a high-level path-planner on the success of adaptive path-planning via cumulative reward and episode length over training steps. The goal-conditioned RL agent reaches higher cumulative reward with lower episode length more quickly than using the goal-conditioned policy alone. This is because goal-conditioned policy can reach nearby goals but often fails and needs assistance from a higher-level planning stage for distant goals.

5 USER STUDY

We conducted a small-scale online behavioral study to validate two human-like adaptive path-planning properties (i.e., identifying shortcuts and performing a detour) of the proposed model. Thirty-three participants (29 men, 4 women, and 0 others) were recruited from a local university (details withheld for anonymity). These participants were between 20 and 36 years of age (mean age = 26.6 and standard deviation = 4.84). Six participants data were not uploaded correctly and hence discarded. In total, we analyzed 27 participant's data for both layouts. For a detailed description of the conducted user study, please refer to Section 4 of the Supplementary Material. On their own computers, participants controlled a 3D virtual agent by applying an egocentric 3D walkthrough motion using a mouse-and-keyboard interface. After a brief training on how to navigate with the mouse and keyboard, participants performed ten training tasks to build a mental representation of the layouts. For each task, participants were asked to find a stationary red balloon hidden in an unknown location in the environment. This location was fixed across all training and testing trials. During the testing phase, participants were then asked to perform five additional navigation tasks. For each of two tasks (i.e., Task 1 and Task 2), participants started from a predefined random location and were asked to locate the hidden balloon using the shortest path. For two other tasks (i.e., Task 3 and Task 4), one of the doors leading to the room containing the red balloon was purposely blocked to simulate a dynamic change to the environment. For the fifth task (i.e., Task 5), a door that had

Table 2: Comparison of average path-distances (in meters) between trajectories generated using simulated agents and human participants on two environments (i.e., Far East Plaza (FEP) and DeepMind layout).

	FEP		DeepMind	
	Agent	Human	Agent	Human
Task 1	35.87 ± 15.50	30.55 ± 9.24	18.12 ± 5.63	22.49 ± 13.60
Task 2	40.53 ± 6.07	37.94 ± 12.22	31.92 ± 6.33	30.38 ± 4.79
Task 3	34.19 ± 6.05	52.16 ± 23.04	45.30 ± 19.10	39.15 ± 8.35
Task 4	30.48 ± 8.94	48.17 ± 19.27	31.37 ± 9.95	36.05 ± 24.07
Task 5	20.52 ± 11.38	30.17 ± 8.84	17.15 ± 2.93	23.55 ± 4.65
Average	**32.33** ± **6.72**	**39.8** ± **8.99**	**28.79** ± **10.37**	**30.32** ± **6.60**

Table 3: We showcase the similarity between humans and agents of their paths that overlapped with the shortest possible paths determined by A* on two environments (i.e., Far East Plaza (FEP) and DeepMind layout).

	FEP		DeepMind	
	Agent	Human	Agent	Human
Task 1	41%	77%	88%	64%
Task 2	48%	38%	34%	57%
Task 3	69%	54%	84%	64%
Task 4	46%	77%	89%	50%
Task 5	89%	38%	98%	64%
Average	**59%**	**57%**	**79%**	**60%**

been blocked for the other training and testing trials was opened to provide a novel shortcut. The order of these five testing tasks was randomized for each participant.

For comparison with the human participants, we used an agent with optimal SR and ARR (i.e., complete knowledge of the environment) that were pre-computed and provided during training to learn a low-level goal-conditioned policy. We generated trajectories from 100 episodes for each of the same five testing tasks using these fully trained optimal agents. In total, 1000 trajectories were generated for two different environments.

5.1 Results

In Figure 1 and 10, we illustrate the trajectories generated by the human participants and optimal agent on a replica of a real-world building (Far East Plaza, Singapore and a simulated environment from DeepMind dataset respectively). In general, the human participants were able to identify the shortest paths in Tasks 1 and 2 and find the goal despite needing to detour in Tasks 3 and 4. Slightly more than half of the participants (64.28%) were able to take the novel shortcut in Task 5. Similarly, the agent consistently found the shortest path and was able to detour around blocked doorways. On average, the agent was able to find shorter path than the human participants (see Table 2).

The longer average path-distance in humans is the result of some participants making an incorrect turn at some decision points and taking a longer route to the goal. This behavior is observed mainly in the FEP layout, which is larger and more complicated than the DeepMind layout. The difference between average path-distances for humans and agents in the DeepMind layout was only 1.53 meters. Moreover, we see substantial similarity between routes taken by

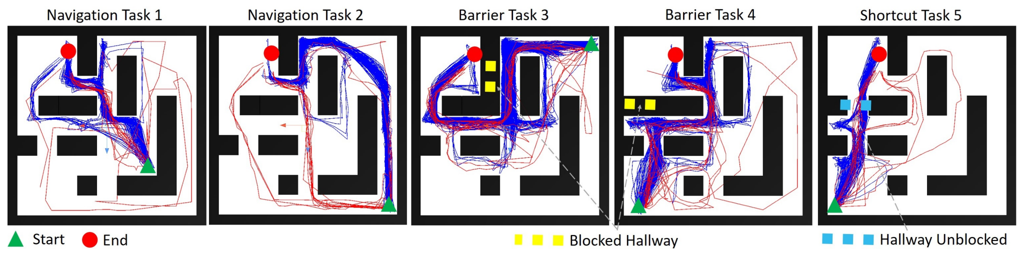

Figure 10: Visualization of trajectories generated by optimal agent and human participants for five wayfinding tasks. Agent's trajectories are visualized in blue and human participants trajectories in red.

agents during all five testing tasks. Indeed, when the agent took an alternate route, it was similar to alternate routes taken by the participants. Our proposed agent can thus produce human-like variation in planning a path between an origin and destination pair. In response to detours in Tasks 3 and 4, we notice that agents took similar paths to reach a goal as humans. In addition, apart from a few agents, the majority of the simulated agents successfully identified shortcuts in Task 5, which was also observed in more than half of the human trajectories. We also generated the shortest possible paths using A* and compared both human and agent trajectories to these shortest possible paths. In Table 3, we note that the agent trajectories were closer to the A* paths than the human trajectories. In summary, our adaptive path-planning agent that combines navigation strategies based on vector navigation and topological graphs successfully negotiated the inserted barrier on 99% of trials. These agents also found novel shortcuts on 99.5% of trials during testing on both complex environments. Finally, a primitive goal vector navigation policy helped the agents to travel to distant goals using an unexplored state space while exploiting novel shortcuts in the absence of any barrier.

5.2 Discussion

One of the critical challenges in the simulation of realistic pedestrian behavior either during evacuation or general circulation is to mimic the decision-making process of humans, which is marred with uncertainty and incomplete knowledge. Classical path-planning algorithms focus on efficiency and path-optimality, which often fails to reproduce human-like wayfinding behavior. The major drawback of such approaches is high computational cost and failure to acknowledge the environmental uncertainty. Moreover, path planning depends entirely on a topological/navigation graph provided as input or computed after dense exploration. The latter will fail to produce a comprehensive navigation graph in sparsely explored environments. One important difference between our method and the classical path-planning approach is identifying nodes/subgoals locations for topological navigation strategy. These subgoals are often at a bottleneck location from which neighboring regions can be efficiently accessed. Our behavioral study provides evidence that the subgoals generated by our method show similarity in a location with the subgoals chosen by human participants. Another important benefit of the proposed method is the reproduction of human-like error in path-planning that results in varied paths between the same origin-destination pair (see Figure 10). We believe that the above-mentioned useful features of our method can be used to simulate realistic pedestrian simulation, increase the navigation believability of non-playable characters in immersive games, and aid in landmark/signage placement during a building design process. Finally, the present work attempts to model human-like navigation by employing the theoretical understanding of the cognitive map supported by neurophysiological evidence in the realm of reinforcement learning (RL).

6 CONCLUSION & FUTURE WORKS

Motivated by the flexibility of human and animal navigation systems, we propose **SNAP** that learns a goal-conditioned policy based on grid cell decoding for vector navigation as a low-level controller in conjunction with a high-level planner over a topological graph formulated by abstracting the state's space. This paper proposes an SR-Entropy-based iterative subgoal discovery and simultaneous goal-conditioned RL policy to solve long-horizon, sparse reward navigation tasks. We showcase via multiple simulations and validation against human data that an efficient selection of state representations for iterative subgoal discovery can finesse the expensive sequential state-state navigation, identify novel shortcuts, perform detours, and adapt behavior during transition revaluations. One of the main limitations of our approach is the manual tuning of edge connection between subgoals during the formulation of the topological graph. We use a user-defined threshold that is sensitive to environment structure. Although our approach is efficient enough to identify bottleneck states quickly, it relies heavily on state sampling to learn SR for state embedding.

ACKNOWLEDGMENTS

The research was partially supported through funding by the Leonhard Obermeyer Center at the Technical University of Munich. The research was also supported in part by NSF awards: IIS-1703883, IIS-1955404, IIS-1955365, RETTL-2119265, and EAGER-2122119.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *arXiv preprint arXiv:1707.01495* (2017).

David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report. Stanford.

Caswell Barry, Colin Lever, Robin Hayman, Tom Hartley, Stephen Burton, John O'Keefe, Kate Jeffery, and Neil Burgess. 2006. The boundary vector cell model of place cell firing and spatial memory. *Reviews in the Neurosciences* 17, 1-2 (2006), 71.

CS Breathnach. 1980. The hippocampus as a cognitive map. *Philosophical Studies* 27 (1980), 263–267.

Simon J Buecher, Christoph Holscher, and Jan Wiener. 2009. Search strategies and their success in a virtual maze. In *Proceedings of the annual meeting of the cognitive science society*, Vol. 31.

Daniel Bush, Caswell Barry, and Neil Burgess. 2014. What do grid cells contribute to place cell firing? *Trends in neurosciences* 37, 3 (2014), 136–145.

Daniel Bush, Caswell Barry, Daniel Manson, and Neil Burgess. 2015. Using grid cells for navigation. *Neuron* 87, 3 (2015), 507–520.

Peter Dayan. 1993. Improving generalization for temporal difference learning: The successor representation. *Neural Computation* 5, 4 (1993), 613–624.

Rohit K Dubey, Samuel S Sohn, Tyler Thrash, Christoph Hoelscher, and Mubbasir Kapadia. 2019. Identifying indoor navigation landmarks using a hierarchical multi-criteria decision framework. In *Motion, Interaction and Games*. 1–11.

Vegard Edvardsen. 2015. A passive mechanism for goal-directed navigation using grid cells. In *Artificial Life Conference Proceedings 13*. MIT Press, 191–198.

Vegard Edvardsen. 2018. Navigating with distorted grid cells. In *Artificial Life Conference Proceedings*. MIT Press, 260–267.

Vegard Edvardsen, Andrej Bicanski, and Neil Burgess. 2020. Navigating with grid and place cells in cluttered environments. *Hippocampus* 30, 3 (2020), 220–232.

Russell A Epstein and Lindsay K Vass. 2014. Neural systems for landmark-based wayfinding in humans. *Philosophical Transactions of the Royal Society B: Biological Sciences* 369, 1635 (2014), 20120533.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. 2019. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint arXiv:1906.05253* (2019).

Ila R Fiete, Yoram Burak, and Ted Brookings. 2008. What grid cells convey about rat location. *Journal of Neuroscience* 28, 27 (2008), 6858–6871.

Samuel J Gershman. 2018. The successor representation: its computational logic and neural substrates. *Journal of Neuroscience* 38, 33 (2018), 7193–7200.

Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. 2005. Microstructure of a spatial map in the entorhinal cortex. *Nature* 436, 7052 (2005), 801–806.

Anna Harutyunyan, Will Dabney, Diana Borsa, Nicolas Heess, Remi Munos, and Doina Precup. 2019. The Termination Critic. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2231–2240.

Zhiao Huang, Fangchen Liu, and Hao Su. 2019. Mapping state space using landmarks for universal goal reaching. *Advances in Neural Information Processing Systems* 32 (2019), 1942–1952.

Leslie Pack Kaelbling. 1993. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the tenth international conference on machine learning*, Vol. 951. 167–173.

Tejas D Kulkarni, Karthik R Narasimhan, Ardavan Saeedi, and Joshua B Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *arXiv preprint arXiv:1604.06057* (2016).

Colin Lever, Stephen Burton, Ali Jeewajee, John O'Keefe, and Neil Burgess. 2009. Boundary vector cells in the subiculum of the hippocampal formation. *Journal of Neuroscience* 29, 31 (2009), 9771–9777.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. 2017a. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2295–2304.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. 2017b. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089* (2017).

Amy McGovern and Andrew G Barto. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density. (2001).

Bruce L McNaughton, Francesco P Battaglia, Ole Jensen, Edvard I Moser, and May-Britt Moser. 2006. Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience* 7, 8 (2006), 663–678.

Ishai Menache, Shie Mannor, and Nahum Shimkin. 2002. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning*. Springer, 295–306.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. 2016. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673* (2016).

Ida Momennejad, Evan M Russek, Jin H Cheong, Matthew M Botvinick, Nathaniel Douglass Daw, and Samuel J Gershman. 2017. The successor representation in human reinforcement learning. *Nature Human Behaviour* 1, 9 (2017), 680–692.

Parham Moradi, Mohammad Ebrahim Shiri, and Negin Entezari. 2010. Automatic skill acquisition in reinforcement learning agents using connection bridge centrality. In *International Conference on Future Generation Communication and Networking*. Springer, 51–62.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296* (2018).

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. 2018. Visual reinforcement learning with imagined goals. *arXiv preprint arXiv:1807.04742* (2018).

Soroush Nasiriany, Vitchyr H Pong, Steven Lin, and Sergey Levine. 2019. Planning with goal-conditioned policies. *arXiv preprint arXiv:1911.08453* (2019).

John O'Keefe and Jonathan Dostrovsky. 1971. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain research* (1971).

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. 2018. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081* (2018).

Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. 2019. Successor options: An option discovery framework for reinforcement learning. *arXiv preprint arXiv:1905.05731* (2019).

Evan M Russek, Ida Momennejad, Matthew M Botvinick, Samuel J Gershman, and Nathaniel D Daw. 2017. Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS computational biology* 13, 9 (2017), e1005768.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *International conference on machine learning*. PMLR, 1312–1320.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Philip Shamash, Sarah F Olesen, Panagiota Iordanidou, Dario Campagner, Banerjee Nabhojit, and Tiago Branco. 2020. Mice learn multi-step routes by memorizing subgoal locations. *BioRxiv* (2020).

Özgür Şimşek and Andrew G Barto. 2004. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*. 95.

Trygve Solstad, Charlotte N Boccara, Emilio Kropff, May-Britt Moser, and Edvard I Moser. 2008. Representation of geometric borders in the entorhinal cortex. *Science* 322, 5909 (2008), 1865–1868.

Trygve Solstad, Edvard I Moser, and Gaute T Einevoll. 2006. From grid cells to place cells: a mathematical model. *Hippocampus* 16, 12 (2006), 1026–1031.

Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. 2017. The hippocampus as a predictive map. *Nature neuroscience* 20, 11 (2017), 1643.

Federico Stella and Alessandro Treves. 2015. The self-organization of grid cells in 3D. *Elife* 4 (2015), e05913.

Martin Stolle and Doina Precup. 2002. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*. Springer, 212–223.

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

Jeffrey S Taube, Robert U Muller, and James B Ranck. 1990. Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *Journal of Neuroscience* 10, 2 (1990), 420–435.

Abhishek Verma and Bérénice Mettler. 2017. Human learning of unknown environments in agile guidance tasks. *arXiv preprint arXiv:1710.07757* (2017).

Chengguang Xu, Christopher Amato, and Lawson LS Wong. 2021. Hierarchical Robot Navigation in Novel Environments using Rough 2-D Maps. *arXiv preprint arXiv:2106.03665* (2021).

Lunjun Zhang, Ge Yang, and Bradly C Stadie. 2020. World Model as a Graph: Learning Latent Landmarks for Planning. *arXiv preprint arXiv:2011.12491* (2020).

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.