2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

TopoNav: Topological Navigation for Efficient Exploration in Sparse Reward Environments

Jumman Hossain[1], Abu-Zaher Faridee[1,2], Nirmalya Roy[1], Jade Freeman[3], Timothy Gregory[3], and Theron Trout[4]

***Abstract*— Autonomous robots exploring unknown environments face a significant challenge: navigating effectively without prior maps and with limited external feedback. This challenge intensifies in sparse reward environments, where traditional exploration techniques often fail. In this paper, we present *TopoNav*, a novel topological navigation framework that integrates active mapping, hierarchical reinforcement learning, and intrinsic motivation to enable efficient goal-oriented exploration and navigation in sparse-reward settings. *TopoNav* dynamically constructs a topological map of the environment, capturing key locations and pathways. A two-level hierarchical policy architecture, comprising a high-level graph traversal policy and low-level motion control policies, enables effective navigation and obstacle avoidance while maintaining focus on the overall goal. Additionally, *TopoNav* incorporates intrinsic motivation to guide exploration towards relevant regions and frontier nodes in the topological map, addressing the challenges of sparse extrinsic rewards. We evaluate *TopoNav* both in the simulated and real-world off-road environments using a Clearpath Jackal robot, across three challenging navigation scenarios: goal-reaching, feature-based navigation, and navigation in complex terrains. We observe an increase in exploration coverage by 7-20%, in success rates by 9-19%, and reductions in navigation times by 15-36% across various scenarios, compared to state-of-the-art methods.**

I. Introduction

Autonomous robot navigation in unknown, unstructured environments poses significant challenges, particularly in the absence of prior maps and reliable localization [1], [2]. In such scenarios, robots must efficiently explore the environment, build accurate representations, and make intelligent decisions to reach their goals, often with limited computational resources and sparse feedback [3], [4]. Traditional approaches, such as simultaneous localization and mapping (SLAM) [5] and sampling-based planning [6], rely heavily on geometric representations and struggle to adapt to the uncertainties and dynamicity of real-world environments. Recent advancements in deep reinforcement learning (RL) have shown promise in enabling robots to learn complex navigation policies directly from raw sensory inputs [7], [8]. However, most RL-based approaches suffer from sample inefficiency, poor generalization to unseen environments, and difficulties in handling sparse reward signals [9]. Hierarchical RL methods [10], [11] attempt to address these issues by learning multi-level policies, but often rely on handcrafted state spaces, pre-defined sub-goals, and task-specific reward shaping [12], limiting their autonomy and adaptability. Topological mapping [13], [14] is a promising approach for efficient navigation in large-scale environments. By representing the environment as a graph of discrete places and their connectivity, topological maps provide a compact and flexible representation that scales well with the size of the environment [15]. However, most topological mapping approaches rely on predefined place recognition methods [16] or assume a fixed set of landmarks [17], making them sensitive to handling ambiguous visual appearances and changes in the environment. They often struggle to adapt to dynamic and unstructured environments, where the appearance of landmarks may change significantly over time.

In this paper, we introduce *TopoNav*, a novel topological navigation framework that integrates active mapping, hierarchical reinforcement learning, and intrinsic motivation to enable efficient and autonomous exploration of unknown environments. *TopoNav* dynamically constructs and maintains a topological map of the environment using a deep neural

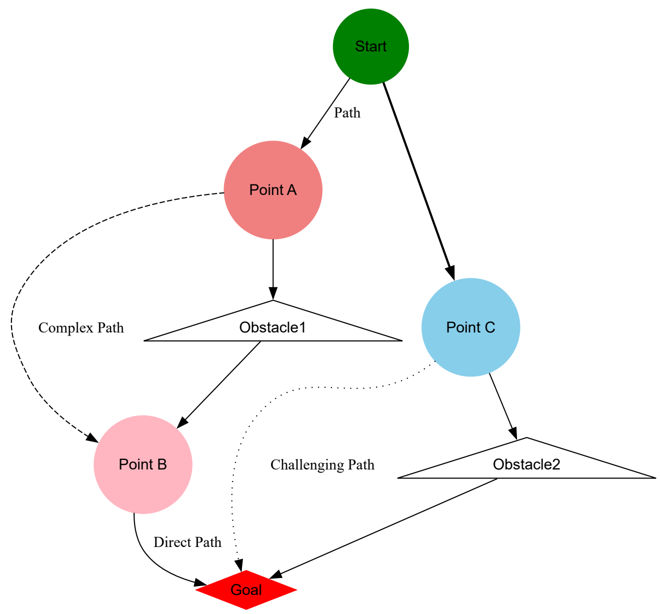


Fig. 1: *TopoNav Navigation Strategies*: The navigation begins at the Start node (green circle) and progresses through designated subgoals—Point A (initial decision point), Point B (complex navigation subgoal), and Point C (alternative challenging subgoal)—toward the Goal (red diamond). The routes illustrate *TopoNav*'s strategy: solid lines represent direct paths to subgoals, a dashed line marks a complex detour around Obstacle1, and a dotted line indicates a potential route for challenging maneuvering near Obstacle2. This diagram shows the robot's strategic navigation from start to finish, highlighting its decision-making and adaptability in outdoor environments with diverse navigational challenges. A real-world scenario is presented in Fig. 4

.

[1]Authors are with the Dept. of Information Systems, University of Maryland, Baltimore County, USA. {jumman.hossain, faridee1, nroy}@umbc.edu
[2]Author is with Amazon Inc. USA. abufari@amazon.com
[3]Authors are with DEVCOM Army Research Lab, USA. {jade.l.freeman2.civ, timothy.c.gregory6.civ}@army.mil
[4]Author is with Stormfish Scientific Corporation. theron.trout@stormfish-sci.com

979-8-3503-7769-9/24/$31.00 ©2024 IEEE

network that learns to extract task-relevant features from raw sensor observations. By combining the strengths of learning-based perception, hierarchical decision-making, and intrinsically motivated exploration, *TopoNav* demonstrates significant improvements in efficiency, robustness, and adaptability compared to state-of-the-art navigation methods.

The key contributions of this work are:

1) **Enhanced Hierarchical Reinforcement Learning with Active Topological Mapping:** We introduce an enhanced hierarchical reinforcement learning framework that extends the Hierarchical Deep Q-Network (H-DQN) [18] architecture by integrating an actively updated topological map and leveraging intrinsic rewards to facilitate multi-level navigation policy learning. The meta-controller is responsible for choosing subgoals from the topological map, while the sub-controllers are designed to reach these subgoals through the execution of primitive actions. This dual strategy ensures that navigation is not only efficient but also directed towards regions of the environment that significantly enhance its understanding. By incorporating an intrinsically motivated learning approach, we effectively address the challenges associated with sparse extrinsic rewards, thereby accelerating the learning of efficient navigation policies.
2) **Dynamic Subgoal Generation and Strategic Landmark Selection:** We design a dynamic subgoal generation mechanism that activates upon detecting landmarks, trees, or objects while navigating. Detected features become part of the topological map as subgoals or nodes, facilitating structured navigation. When multiple landmarks are detected at similar distances, *TopoNav* utilizes a strategic landmark selection strategy. This approach gives priority to landmarks that are most informative and relevant, considering their novelty and alignment with the final goal. This method promotes efficient exploration and ensures navigation is goal-oriented, even when presented with numerous potential subgoals.
3) **Experimental Validation of Superior Performance:** We extensively evaluate *TopoNav* in diverse simulated environments and real-world scenarios, benchmarking against state-of-the-art baselines. It showcases a significant increase in exploration coverage (7-20%), and navigation success rates (9-19%), and achieves substantial reductions in navigation times (15-36%) across various scenarios. These improvements demonstrate *TopoNav*'s superior navigation in complex environments.

The rest of the paper is organized as follows. Section II discusses related work on robot navigation, topological mapping, and reinforcement learning. The background and problem formulation are discussed in Section III. Section IV presents the *TopoNav* framework. Section V describes the experimental setup, environments, and evaluation metrics. Section V presents the results and analysis of the experiments, comparing *TopoNav* with SOTA navigation methods. Finally, Section VI concludes the paper and discusses future research directions.

II. Related Work

This section discusses prior works related to autonomous robot navigation, focusing on learning-based methods, and topological mapping techniques.

A. Learning-Based Navigation

Recent advancements in deep reinforcement learning (RL) have led to the development of learning-based navigation methods that can learn effective policies directly from sensory inputs. Deep Q-Networks (DQN) [19] have been applied to navigation tasks, enabling robots to learn collision-free paths in different environments. However, these methods typically require a large amount of training data and may struggle to generalize to unseen environments. Hierarchical reinforcement learning (HRL) methods have been proposed to address the challenges of sparse rewards and long-horizon tasks in navigation [10], [18], [20]. These approaches typically learn a hierarchy of policies, where higher-level policies select subgoals or actions for lower-level policies to execute. However, most HRL methods [21] assume access to structured representations of the environment or rely on pre-defined subgoals, limiting their applicability in unknown and unstructured environments.

B. Topological Mapping Navigation

Topological mapping techniques represent the environment as a graph, where nodes correspond to distinct places and edges represent navigable paths between them [13], [14]. These methods focus on capturing the connectivity and adjacency information of the environment rather than maintaining a precise metric map. Spectral clustering algorithms have been used to construct topological maps from sensor data [15], [22]. These methods exploit the eigenstructure of the similarity matrix to partition the environment into distinct regions. However, they often require a pre-specified number of clusters and may not adapt well to changes in the environment. Incremental topological mapping approaches incrementally build and refine the topological map as the robot explores the environment [23], [24]. These methods typically use appearance-based place recognition techniques to detect loop closures and update the map accordingly. However, they may struggle in environments with significant changes in appearance. Recently, learning-based approaches have been proposed for topological mapping and localization [25], [26]. These methods learn to extract topological representations directly from sensory inputs using deep neural networks. Suomela et al. [27] proposed PlaceNav, a topological navigation approach that utilizes visual place recognition for subgoal selection. Wiyatno et al. [28] proposed a lifelong topological navigation approach that builds and continuously refines a sparse topological graph. However, they often require a large amount of training data and may not generalize well to unseen and sparse reward environments.

While existing topological navigation methods have shown promising results, they often rely on dense reward signals or extensive exploration to build effective representations of the environment. In contrast, *TopoNav* is specifically designed to operate in sparse-reward settings by incorporating intrinsic motivation and hierarchical reinforcement learning, enabling efficient navigation and map construction with limited extrinsic feedback. By dynamically constructing and refining the topological map during exploration, *TopoNav* can adapt to changes in the environment. The hierarchical policy architecture allows for effective navigation and obstacle avoidance, while intrinsic motivation guides exploration towards informative regions.

III. Background and Problem Formulation

In this section, we outline the topological mapping for navigation, the Hierarchical Deep Q-Networks (H-DQN) for policy learning, and our problem formulation.

A. Topological Mapping

Topological mapping represents the environment as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $v \in \mathcal{V}$ correspond to distinct places or landmarks, and edges $e \in \mathcal{E}$ represent the connectivity between them. This representation allows for efficient path planning and navigation by abstracting away the metric details and focusing on the high-level structure of the environment [15]. In our approach, we dynamically construct and update the topological map $\mathcal{M}$ as the robot explores the environment. Each node $v_i \in \mathcal{V}$ is associated with a feature vector $\mathbf{f}_i$ that encodes the sensory observations at that location. The edges $e_{ij} \in \mathcal{E}$ represent the traversability between nodes v_i and v_j, which can be determined based on the robot's motion model and the observed environmental conditions. The topological map $\mathcal{M}$ is used to guide the exploration process and enable efficient decision-making. By representing the environment as a graph, the robot can plan paths between different nodes and make high-level decisions based on the connectivity of the map. This allows for more strategic exploration and navigation compared to purely reactive approaches.

B. Hierarchical Deep Q-Networks (H-DQN)

Hierarchical Deep Q-Networks (H-DQN) [18] is a hierarchical reinforcement learning algorithm that extends the standard Deep Q-Networks (DQN) [19]. We choose the H-DQN due to its efficiency in managing tasks across different abstraction levels and its enhanced sample efficiency through off-policy learning and experience replay, which is crucial for real-world robotics applications where data collection can be costly and time-consuming. H-DQN learns a two-level policy, the high-level policy (meta-controller) learns to select subgoals from the topological map, while the low-level policies (sub-controllers) learn to generate actions to reach the selected subgoals. The meta-controller is represented by a Q-network $Q^{\mu}(s, g; \theta^{\mu})$, where $s \in \mathcal{S}$ is the current state, $g \in \mathcal{G}$ is a subgoal, and θ^{μ} are the network parameters. The meta-controller selects subgoals based on the learned Q-values, which estimate the expected cumulative reward for reaching each subgoal. The sub-controllers are represented by a set of Q-networks $Q^{g}(s, a; \theta^{g}) | g \in \mathcal{G}$, where $s \in \mathcal{S}$ is the current state, $a \in \mathcal{A}$ is an action, and θ^{g} are the network parameters for subgoal g. Each sub-controller learns a policy to navigate from the current state to the corresponding subgoal by selecting actions that maximize the learned Q-values. The meta-controller and sub-controllers are trained simultaneously with separate replay buffers and target networks for each level of the hierarchy.

C. Problem Formulation

We formulate the problem as a Markov Decision Process (MDP) with a hierarchical structure. The MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability function, $\mathcal{R}$ is the reward function, and γ is the discount factor. The state space $\mathcal{S}$ consists of the robot's sensory observations and the current topological map $\mathcal{M}$. The action space $\mathcal{A}$ is hierarchically structured, with high-level actions (subgoals) $\mathcal{A}_h$ selected by the meta-controller and low-level actions (primitives) $\mathcal{A}_l$ executed by the controller. The transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ determines the next state based on the current state and the executed action. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is sparse, providing positive rewards for reaching the goal or completing milestones, and intrinsic rewards for exploring new nodes or discovering new connections in the topological map. The objective is to learn a hierarchical policy $\pi = (\pi_h, \pi_l)$ that maximizes the expected cumulative reward:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right], \tag{1}$$

where π^* is the optimal policy, $\pi_h : \mathcal{S} \rightarrow \mathcal{A}_h$ is the high-level policy (meta-controller), $\pi_l : \mathcal{S} \times \mathcal{A}_h \rightarrow \mathcal{A}_l$ is the low-level policy (controller), r_t is the reward at time step t, and γ is the discount factor. In this context, the state space $\mathcal{S}$ includes a dynamically constructed topological map $\mathcal{M}$, where nodes represent automatically identified landmarks. The high-level action space $\mathcal{A}_h$ consists of subgoals, allowing the meta-controller to make decisions based on the evolving map. This approach enables efficient navigation in unknown environments without relying on predefined waypoints.

IV. TopoNav: Topological Map Generation and Navigation

In this section, We explain the major stages of the *TopoNav* approach. Fig. 2 shows how different modules in our method are connected.

A. Attention-based Feature Detection

TopoNav integrates an attention-based feature detection module, combining a ResNet-50 CNN [29] with a Convolutional Block Attention Module (CBAM) [30], to detect landmarks, objects, and trees from RGB images $\mathbf{I}_i \in \mathbb{R}^{H \times W \times 3}$, where H and W are the height and width of the input image, respectively. This module processes images to produce feature maps $\mathbf{F}_i \in \mathbb{R}^{h \times w \times c}$ (h and w are the height and width of the feature map, and c is the number of channels) refining them through CBAM to emphasize relevant features for landmark detection. Utilizing a sliding window approach on the refined feature map $\mathbf{F}'_i$, *TopoNav* identifies potential Regions of Interest (ROIs) corresponding to navigational markers. Each ROI's feature vector $\mathbf{f}_{ROI} \in \mathbb{R}^d$, obtained through ROI pooling, undergoes a binary classification to ascertain the presence of valid landmarks, objects, or trees, assigning a probability score $\mathbf{P}_{ROI} \in [0, 1]$. For each detected landmark, object, or tree, *TopoNav* extracts its corresponding feature vector $\mathbf{f}_{ROI}$ and uses it to create a new node or subgoal in the topological map.

B. Subgoal Generation and Navigation

In *TopoNav*, subgoals or nodes are generated dynamically based on the landmarks, trees, or objects encountered by the robot while navigating towards the final goal. Whenever a new landmark or object is detected, it is compared with the existing nodes in the topological map using a similarity measure (described in Section IV-C). If the similarity is below a threshold, the landmark is added as a new subgoal or node to the topological map. However, when multiple landmarks are detected at similar distances, *TopoNav* employs

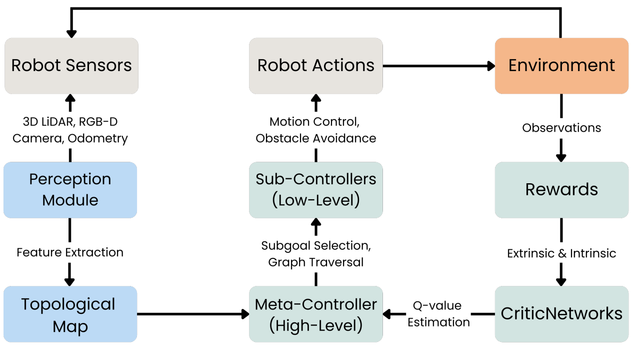


Fig. 2: Overview of *TopoNav* System Architecture.

a selection strategy to prioritize the most informative and relevant landmark as the next subgoal based on novelty and goal-directedness. The novelty of a landmark l is calculated based on the number of previous visits, using a novelty factor that decreases exponentially with the number of visits:

$$N(l) = e^{-\lambda \cdot \text{visits}(l)} \tag{2}$$

where λ is a decay parameter controlling the rate at which the novelty factor decreases with visits. The goal-directedness of a landmark is calculated as the cosine similarity between the direction vector $\mathbf{v}_l$ towards the landmark l and the direction vector towards the goal $\mathbf{v}_g$. The landmark with the highest score is selected as the next subgoal. The weights w_N and w_{GD} adjust the importance of novelty and goal-directedness in landmark selection, balancing exploration, and direct goal attainment. (See Algorithm 1)

$$GD(l) = \frac{\mathbf{v}_l \cdot \mathbf{v}_g}{||\mathbf{v}_l|| \cdot ||\mathbf{v}_g||} \tag{3}$$

Algorithm 1: Strategic Landmark Selection

Input: Detected landmarks L, current position $\mathbf{p}$, goal position $\mathbf{g}$
Output: Selected landmark l_{best}

Initialize max_score $= -\infty$
foreach *landmark* $l \in L$ **do**
  $N(l) = e^{-\lambda \cdot \text{visits}(l)}$
  $\mathbf{v}_l = \text{direction\_vector}(\mathbf{p}, l)$
  $\mathbf{v}_g = \text{direction\_vector}(\mathbf{p}, \mathbf{g})$
  $GD(l) = \frac{\mathbf{v}_l \cdot \mathbf{v}_g}{||\mathbf{v}_l|| \cdot ||\mathbf{v}_g||}$
  $\text{score}(l) = w_N \cdot N(l) + w_{GD} \cdot GD(l)$
  if *score*$(l) >$ *max_score* **then**
    max_score $= \text{score}(l)$
    $l_{\text{best}} = l$
return l_{best}

During navigation, the meta-controller selects the next subgoal based on the current state and the robot's overall goal. If no landmarks are detected within a certain distance threshold, *TopoNav* generates a new subgoal along the robot's current trajectory at the maximum detection range to ensure continuous progress towards the final goal. The sub-controller's policy generates a sequence of primitive actions or low-level skills to navigate the robot towards the selected subgoal, aiming to maximize the expected cumulative reward. The navigation

Algorithm 2: TopoNav: Topological Map Generation and Navigation

Input: G_g, the global end goal; d_{thresh}, distance threshold for subgoal generation
Output: T_m, the topological map with nodes (N) and edges (E)

Initialize meta-controller Q^μ and sub-controllers Q^g.
Initialize replay buffers $\mathcal{D}^\mu, \mathcal{D}^g$ for each subgoal g.
$T_m \leftarrow \emptyset$; $s_0 \leftarrow$ GetInitialState(); $g_0 \leftarrow G_g$.
while $g_t \neq G_g$ **do**
  $\mathcal{L} \leftarrow \text{DetectLandmarks}(s_t)$.
  if *len*$(\mathcal{L}) > 0$ **then**
    $l_{\text{best}} \leftarrow$ Use Algorithm 1$(\mathcal{L}, s_t, G_g)$.
    if *IsNewNode*(l_{best}) **then**
      Add l_{best} to T_m; $g_t \leftarrow l_{\text{best}}$.
  else
    if *dist*$(s_t, g_t) > d_{thresh}$ **then**
      $g' \leftarrow$ Generate new subgoal; Add g' to T_m; $g_t \leftarrow g'$.
  if $g_t \neq G_g$ **then**
    $a_t \leftarrow$ Select action;
    Execute a_t; Observe s_{t+1}, r_t; Store transition.
    if *reached* g_t **then**
      $g_t \leftarrow G_g$.
  else
    Update subgoal; Store and update transition.
  $s_t \leftarrow s_{t+1}$.
return T_m

process continues until the robot reaches the final goal or a maximum number of steps is exceeded. (See Algorithm 2)

C. Reward Structure

To address the challenges of sparse reward environments, *TopoNav* employs a carefully designed reward structure that combines sparse extrinsic rewards, dense intrinsic rewards, and penalties for suboptimal exploration behavior. The total reward r received by the agent at each time step is calculated as follows:

$$\begin{aligned} r = {} & \alpha \cdot r_{ex} + \beta \cdot (r_{in} + r_{sg} + r_{fe} + r_{ep} + r_{ue}) \\ & + \gamma \cdot (r_p + r_{sd} + r_{te} + r_{ob}) \end{aligned} \tag{4}$$

where r_{ex} represents the sparse extrinsic reward, which is given only when the agent reaches the final goal (R_{goal}) or completes a milestone ($R_{milestone}$):

$$r_{ex} = \begin{cases} R_{goal}, & \text{if } s_{t+1} \text{ is the final goal} \\ R_{milestone}, & \text{if } s_{t+1} \text{ completes a milestone} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The intrinsic rewards r_{in}, r_{sg}, r_{fe}, r_{ep}, and r_{ue} encourage exploration and map expansion. r_{in} is based on the novelty of the visited states, calculated as the inverse square root of the visitation count:

$$r_{in} = \frac{1}{\sqrt{N(s_t)}} \tag{6}$$

r_{sg} is a constant reward given when the agent discovers a new subgoal. The frontier exploration reward r_{fe} encourages the agent to prioritize the exploration of frontier nodes in the topological map:

$$r_{fe} = \lambda_{fe} \cdot \frac{N_{fn}}{N_{tn}} \tag{7}$$

where N_{fn} is the number of new frontier nodes reached, N_{tn} is the total number of nodes in the topological map, and λ_{fe} is a scaling factor. The exploration progress reward r_{ep} rewards the agent for increasing the explored area or adding new nodes to the topological map:

$$r_{ep} = \lambda_{ep} \cdot \frac{\Delta A}{A_{total}} \tag{8}$$

where ΔA is the increase in the explored area, A_{total} is the total area of the environment, and λ_{ep} is a scaling factor. The uncertainty-driven exploration reward r_{ue} encourages the agent to explore regions where its knowledge is limited:

$$r_{ue} = \lambda_{ue} \cdot U(s) \tag{9}$$

where $U(s)$ is the uncertainty estimate of the agent's knowledge about the current state s, and λ_{ue} is a scaling factor. To discourage suboptimal exploration behavior, *TopoNav* incorporates several penalty terms. The penalty for revisiting states r_p discourages the agent from excessively revisiting previously explored areas:

$$r_p = -\lambda_p \cdot \frac{N(s_t) - 1}{\sqrt{N(s_t)}} \tag{10}$$

where $N(s_t)$ is the number of times the state s has been visited, and λ is a scaling factor that controls the magnitude of the penalty. The term $\frac{N(s_t)-1}{\sqrt{N(s_t)}}$ ensures that the penalty increases as the number of revisits grows, but with a diminishing effect to avoid completely discouraging revisits. The subgoal diversity penalty r_{sd} penalizes the agent for selecting subgoals that are similar to previously visited subgoals:

$$r_{sd} = -\lambda_{sd} \cdot \max_{g_h \in \mathcal{H}} \text{sim}(g_t, g_h) \tag{11}$$

where g_t is the current subgoal, $\mathcal{H}$ is the history of visited subgoals, sim$(\cdot)$ is a similarity function, and λ_{sd} is a scaling factor. In *TopoNav*, the similarity function sim$(\cdot)$ is based on the Euclidean distance between the feature vectors associated with the subgoals. Each subgoal g is represented by a feature vector f_g that encodes its visual and spatial characteristics. The similarity between two subgoals g_t and g_h is calculated as:

$$\text{sim}(g_t, g_h) = \sqrt{\sum_{i=1}^{n} (f_{g_t}^{(i)} - f_{g_h}^{(i)})^2} \tag{12}$$

This similarity measure allows *TopoNav* to quantify the dissimilarity between subgoals based on their visual and spatial attributes, promoting the selection of diverse subgoals during exploration. The temporal exploration penalty r_{te} penalizes the agent for long periods of non-exploration:

$$r_{te} = -\lambda_{te} \cdot (t - t_{last_exp}) \tag{13}$$

where t is the current time step, t_{last_exp} is the time step of the last exploration progress, and λ_{te} is a scaling factor. An additional penalty r_{ob} is introduced to penalize the agent for hitting obstacles:

$$r_{ob} = \begin{cases} R_{obstacle}, & \text{if the agent hits an obstacle} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $R_{obstacle}$ is a negative constant reward. This penalty encourages the agent to avoid collisions and navigate safely in the environment. The hyperparameters α, β, and γ balance the contributions of the extrinsic rewards, intrinsic rewards, and penalties, respectively.

D. Hierarchical Policy Learning

TopoNav utilizes the H-DQN algorithm to develop navigation policies across different abstraction levels. This framework employs a meta-controller for high-level subgoal selection and sub-controllers for detailed navigation tasks. The meta-controller, leveraging an attention-based feature vector $\mathbf{f}_t$ from current observations, selects subgoals from the topological map using a DQN-based policy $\pi_m(s_m, g; \theta_m)$. This selection process prioritizes key locations or landmarks in the environment, informed by both the robot's current state and the desired goal features. On the other hand, sub-controllers focus on reaching these subgoals by executing actions derived from a separate DQN policy $\pi_s(s_s, a; \theta_s)$, guided by the local obstacle map and the robot's kinematics to ensure safe navigation. The carefully designed reward structure (described in section IV-C) plays an important role in this hierarchy. Training alternates between meta and sub-controllers using their respective experiences and policies, optimizing a composite reward structure that balances exploration, safety, and goal orientation. This approach enables *TopoNav* to adaptively learn efficient navigation strategies in sparse-reward settings, leveraging the strengths of hierarchical learning and attention-based feature extraction.

Theorem 1 (Subgoal Reachability). *Given a topological map $G = (V, E)$ constructed by TopoNav, with V representing nodes (subgoals) and E representing edges (navigable paths), for any two nodes $v_i, v_j \in V$, if a path exists from v_i to v_j in G, then the sub-controller policy π_s learned by TopoNav can navigate the robot from v_i to v_j with a probability of at least $1-\epsilon$. Here, ϵ is a small positive constant representing the upper bound of navigation failure probability, which accounts for environmental stochasticity and the convergence of the learning algorithm. TopoNav minimizes ϵ through robust feature extraction and continuous map refinement.*

Proof. Consider a path $P_{ij} = \{v_i, e_{i1}, v_1, \ldots, e_{(k-1)j}, v_j\}$ between nodes v_i and v_j in the topological map G, incorporating both nodes and the edges $e_{pq} \in E$ between them. The sub-controller policy π_s, leveraging DQN methodologies, aims for a navigation success across each edge e_{pq} with a probability $\geq 1 - \epsilon_{pq}$, where ϵ_{pq} signifies the maximum probability of failing to navigate from v_p to v_q successfully. The overall probability of navigating the entire path P_{ij}, as the product of the success probabilities for its constituting edges, is $\prod_{e_{pq} \in P_{ij}} (1 - \epsilon_{pq})$. This is theoretically at least $1 - \epsilon$, where $\epsilon = k \max_{p,q} \epsilon_{pq}$ represents the compounded probability of any failure occurring along the path with k edges. Therefore, with the sub-controller policy π_s, *TopoNav* can proficiently facilitate navigation from v_i to v_j within the given environmental conditions and learning algorithm constraints, maintaining a success probability of at least $1 - \epsilon$. □

V. Experiments and Results

In this section, we present the experimental setup, results, and analysis of the *TopoNav* framework. We evaluate the performance of *TopoNav* in both simulated and real-world environments and compare it with state-of-the-art baselines.

A. Evaluation Environments

The *TopoNav* framework is evaluated in a range of simulated and real-world environments. The simulated environments are created using the Unity 3D engine and include outdoor urban scenes and off-road terrains. The simulated environments incorporate landmarks and objects similar to those in our real-world test scenarios (Fig. 3d). The real-world experiments are conducted in an outdoor off-road environment using a Clearpath Jackal robot. The environments are designed to cover various navigation challenges, such as narrow passages, obstacles, uneven terrain, and dead ends. The size of the environments ranges from 20m × 20m to 200m × 200m, and the complexity is varied by adjusting the density and arrangement of obstacles. For each environment, multiple starting and goal locations are randomly sampled to create a diverse set of navigation tasks. The performance of *TopoNav* is evaluated over 100 episodes in each environment, and the average metrics are reported.

B. Evaluation Metrics

We evaluate the performance of *TopoNav* and the baselines using the following metrics:

- **Success Rate**: The percentage of successful navigation episodes where the robot reaches the goal location within a specified time limit.
- **Navigation Time**: The average time taken by the robot to reach the goal location in successful episodes.
- **Trajectory Length**: The average length of the robot's trajectory in successful episodes.
- **Exploration Coverage**: The percentage of the environment explored by the robot during navigation.

C. Baselines

To rigorously evaluate *TopoNav*'s performance, we benchmark it against a diverse set of state-of-the-art baselines:

- **PlaceNav**: A topological navigation approach that utilizes visual place recognition for subgoal selection and integrates a Bayesian filter to improve the temporal consistency of subgoal selection [27].
- **TopoMap**: A method focusing on navigation through the construction of topological maps from predefined landmarks [15], providing a direct comparison to assess the benefits of *TopoNav*'s dynamic mapping and navigation strategy.
- **ViNG (Visual Navigation with Goals)**: A learning-based navigation approach that learns to navigate to visual goals in open-world environments by combining hierarchical planning and topological mapping [31].
- **Lifelong Topological Visual Navigation (LTVN)**: A topological navigation method that builds and continuously refines a sparse topological graph [28].

D. Implementation Details

Our RL network is implemented in PyTorch and trained using simulated terrains with a Clearpath Husky robot in ROS Noetic and Unity Simulation framework, which allows for testing in realistic outdoor environments with diverse terrain and objects. The simulated Husky robot is mounted with a Velodyne VLP16 3D LiDAR. The network is trained in a workstation with a 10th-generation Intel Core i9-10850K processor and an NVIDIA GeForce RTX 3090 GPU. During training, our network uses a batch size of 128 and performs gradient updates using the Adam optimizer with learning rate $\lambda = 10^{-4}$. The training process occurs over 1 million steps in our simulation environment, with performance evaluated at regular intervals. For real-time deployment and inference, we use the Jackal UGV from Clearpath Robotics equipped with a 3D VLP-32C Velodyne Ultrapuck LiDAR and an AXIS Fixed IP Camera. It also includes an onboard Intel computer system, equipped with an Intel i7-9700TE CPU and an NVIDIA GeForce GTX 1650 Ti GPU. We utilize the LiDAR sensor provided 3D point cloud data for obstacle detection and avoidance using the Dynamic Window Approach (DWA) [32].

Metrics	Methods	Scenario 1	Scenario 2	Scenario 3
Success Rate (%)	PlaceNav	89	87	88
	TopoMap	82	79	80
	ViNG	84	82	83
	LTVN	90	88	89
	TopoNav (Ours)	**98**	**94**	**92**
Navigation Time (s)	PlaceNav	37.5	38.2	39.0
	TopoMap	47.5	46.9	48.2
	ViNG	39.8	40.5	41.2
	LTVN	36.2	37.0	37.8
	TopoNav (Ours)	**30.6**	**33.8**	**35.1**
Trajectory Length (m)	PlaceNav	13.8	14.2	14.5
	TopoMap	15.8	15.2	15.7
	ViNG	14.4	14.9	15.3
	LTVN	12.9	13.5	13.8
	TopoNav (Ours)	**10.4**	**11.6**	**12.3**
Exploration Coverage (%)	PlaceNav	83	81	82
	TopoMap	75	73	74
	ViNG	79	77	78
	LTVN	84	83	85
	TopoNav (Ours)	**90**	**88**	**89**

TABLE I: Comparative Analysis: *TopoNav* vs. SOTA methods across scenarios, highlighting performance in three distinct environments (Section V-E).

E. Testing Scenario

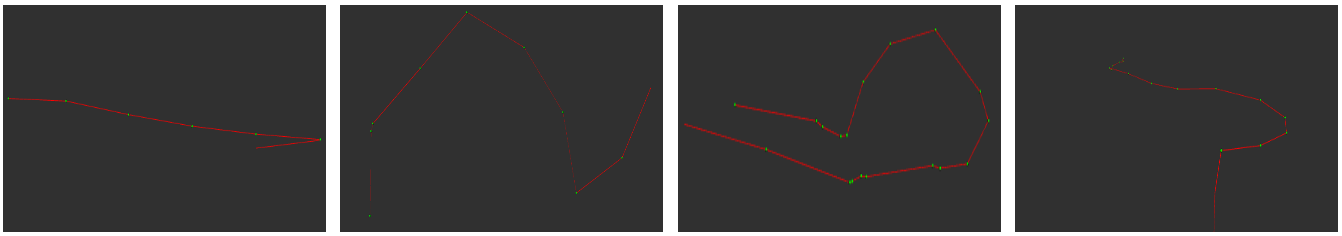
We evaluate our topology based navigation framework in three scenarios (see Fig. 3).

- **Scenario 1 (Goal Reaching):** In this scenario, the robot is placed in an obstacle-free outdoor environment with an objective of reaching a given goal location.
- **Scenario 2 (Feature-Based Navigation):** In this scenario, the robot is placed in an outdoor area with static natural features such as trees, rocks, structures, and sharp hills. The goal is to reach the destination using the natural features for localization and mapping, challenging its environmental perception and landmark utilization.
- **Scenario 3 (Navigating on Complex Terrains):** In this scenario, the robot is placed in an environment with both obstacles and landmarks, this scenario assesses the robot's simultaneous obstacle avoidance and feature-based navigation, testing its adaptability in complex settings.

F. Performance Comparison and Analysis

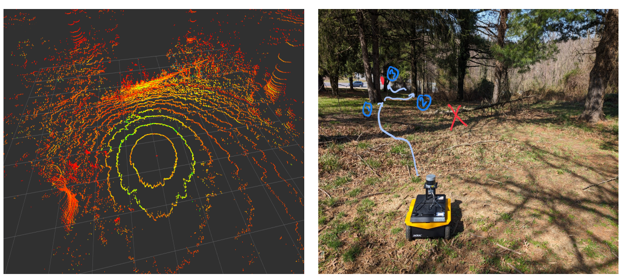
Table I demonstrates *TopoNav*'s significant improvements over existing navigation systems. Our framework outperforms baseline methods across key performance metrics, highlighting its effectiveness in real-world scenarios. In goal-reaching tasks, *TopoNav* achieves a 98% success rate, surpassing PlaceNav [27], LTVN [28], and ViNG [31]. These baselines often

(a) (b) (c) (d)



(e) (f) (g) (h)

Fig. 3: (a-d) The robot navigates through various outdoor environments: (a) an open space without obstacles/vegetation (Scenario 1), (b) a natural setting with trees/vegetation (Scenario 2), (c) a cluttered environment with obstacles and landmarks (Scenario 3), and (d) a diverse terrain in simulation. (e-h) The corresponding topological maps generated by *TopoNav* for each scenario. The topological maps capture the connectivity and traversability of the environments, representing key locations and paths. The green dots in the maps represent the nodes or subgoals, which correspond to landmarks or distinct places. The edges (red lines) indicate the navigability between these nodes, enabling efficient path planning and navigation for the robot in each setting.



(a) (b)

Fig. 4: (a) A 3D point cloud representation of the outdoor environment obtained from the robot's LiDAR sensor, showcasing the terrain and obstacle details. (b) Robot navigating through a physical environment, guided by the *TopoNav* framework, executes an immediate obstacle avoidance maneuver to select an alternative route when faced with an obstruction (red cross, fallen trees).

struggle with efficient path planning in sparsely rewarded environments. In contrast, *TopoNav* leverages its hierarchical structure and dynamic topological mapping for more effective route optimization. The limitations of the baselines become evident in feature-based navigation, where they fail to consistently utilize natural landmarks for navigation. *TopoNav* excels in this scenario, achieving an 94% success rate. This demonstrates its superior ability to accurately navigate by effectively using environmental features for guidance. The complex terrain scenario further highlights *TopoNav*'s robustness. With a 92% success rate, it significantly outperforms baseline systems, which often experience performance degradation in challenging navigational conditions. *TopoNav*'s adaptability to diverse and unpredictable terrains underscores the shortcomings of less flexible systems that heavily depend on stable and predictable environments. Moreover, the integration of CBAM [30] into *TopoNav*'s feature extraction module greatly enhances landmark detection, a common shortcoming in baseline methods. This results in an average 8% increase in detection accuracy, improving the reliability of subgoal generation and map construction. The subsequent impact on navigation success—up to a 14% improvement—and reduction in navigation times—up to a 28% decrease—further establishes *TopoNav* as a significant advancement in autonomous robot navigation.

Method	Success Rate (%)	Navigation Time (s)	Trajectory Length (m)	Exploration Coverage (%)
TopoNav	**98**	**30.6**	**10.4**	**90**
TopoNav w/o Topo Map	88	40.2	14.1	80
TopoNav w/o Hierarchy	85	45.3	15.7	75
TopoNav w/o Attention	86	42.5	14.8	78

TABLE II: Ablation study results.

G. Ablation Study

To analyze the contribution of each component in *TopoNav*, we conduct an ablation study by removing the topological mapping, the hierarchical structure, and the attention-based CNN separately. Table II shows the results of the ablation study in the outdoor environment (Scenario 1 V-E). The results show that removing any component from *TopoNav* decreases success rates and exploration coverage while increasing navigation times and trajectory lengths.

VI. Conclusion, Limitations, and Future Directions

In this paper, we introduced *TopoNav*, a novel framework for autonomous navigation in unknown environments with sparse rewards. *TopoNav* integrates hierarchical reinforcement learning with dynamic topological mapping to enable efficient exploration, map construction, and goal-directed navigation. The key components of *TopoNav* include a two-level hierarchical policy learning architecture, an active topological mapping approach, and an intrinsic reward mechanism for encouraging exploration. Despite the promising results, *TopoNav* has several limitations that provide an opportunity for future research.

One limitation is the scalability of *TopoNav* to larger and more complex environments. The current implementation may face computational challenges when dealing with extensive topological maps and long-horizon navigation tasks. Future work could explore techniques for map compression and distributed computing to improve the scalability of *TopoNav*. Furthermore, developing techniques for distributed map construction, information sharing, and coordinated decision-making could enable more efficient exploration and navigation in large-scale environments with multiple robots.

ACKNOWLEDGMENT

This work has been partially supported by the DEVCOM Army Research Laboratory (ARL) under a cooperative agreement (W911NF2120076), ONR Grant #N00014-23-1-2119, NSF CAREER Award #1750936, NSF REU Site Grant #2050999, and NSF CNS EAGER Grant #2233879.

REFERENCES

[1] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[2] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[3] Mitchell Chen, Alec Radford, Ilya Sutskever, Ilya Sutskever, and Dario Amodei. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.

[4] Fatemeh Niroui, Kaicheng Zhang, Zepeng Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. volume 4, pages 610–617. IEEE, 2019.

[5] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[6] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.

[7] Lei Tai, Qiong Zhang, Meng Liu, Joschka Boedecker, and Wolfram Burgard. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. *arXiv preprint arXiv:1703.00420*, 2017.

[8] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[9] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15246–15257, 2019.

[10] Andrew Levy, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations*, 2019.

[11] Amy Zhang, Adam Lerer, Sainbayar Sukhbaatar, Rob Fergus, and Arthur Szlam. Generating adjacency-constrained subgoals for hierarchical reinforcement learning. *arXiv preprint arXiv:2006.11485*, 2020.

[12] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 31:3303–3313, 2018.

[13] Benjamin Kuipers. The spatial semantic hierarchy. *Artificial intelligence*, 119(1-2):191–233, 2000.

[14] Jaime Boal, Alvaro Sanchez-Miralles, and Alvaro Arranz. Topological simultaneous localization and mapping: a survey. *Robotica*, 32(5):803–821, 2014.

[15] Fabian Blochliger, Marius Fehr, Marcin Dymczyk, Thomas Schneider, and Rol Siegwart. Topomap: Topological mapping and navigation based on visual slam maps. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3818–3825. IEEE, 2018.

[16] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

[17] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

[18] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29:3675–3683, 2016.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. In *Nature*, volume 518, pages 529–533. Nature Publishing Group, 2015.

[20] Siyuan Li, Rui Wang, Minxue Tang, and Chongjie Zhang. Hierarchical reinforcement learning with advantage-based auxiliary rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[21] Chengshu Li, Fei Xia, Roberto Martin-Martin, and Silvio Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pages 603–616. PMLR, 2020.

[22] Yugang Liu and Goldie Nejat. Robotic urban search and rescue: A survey from the control perspective. volume 72, pages 147–165. Springer, 2013.

[23] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[24] Viorela Ila, Lukas Polok, Marek Solony, and Pavel Svoboda. Slam++-a highly efficient and temporally scalable incremental slam framework. *The International Journal of Robotics Research*, 36(2):210–230, 2017.

[25] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations*, 2020.

[26] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12875–12884, 2020.

[27] Lauri Suomela, Jussi Kalliola, Harry Edelman, and Joni-Kristian Kämäräinen. Placenav: Topological navigation through place recognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[28] Rey Reza Wiyatno, Anqi Xu, and Liam Paull. Lifelong topological visual navigation. *IEEE Robotics and Automation Letters*, 7(4):9271–9278, 2022.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[31] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Ving: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021.

[32] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.