Rapid Exploration for Open-World Navigation with Latent Goal Models

Anonymous Author(s)

Abstract: We describe a robotic learning system for autonomous exploration and navigation in diverse, open-world environments. At the core of our method is a learned latent variable model of distances and actions, along with a non-parametric topological memory of images. We use an information bottleneck to regularize the learned policy, giving us (i) a compact visual representation of goals, (ii) improved generalization capabilities, and (iii) a mechanism for sampling feasible goals for exploration. Trained on a large offline dataset of prior experience, the model acquires a representation of visual goals that is robust to task-irrelevant distractors. We demonstrate our method on a mobile ground robot in open-world exploration scenarios. Given an image of a goal that is up to 80 meters away, our method leverages its representation to explore and discover the goal in under 20 minutes, even amidst previously-unseen obstacles and weather conditions.

1 Introduction

Robustness is a key challenge in learning to navigate diverse, real-world environments. A robotic learning system must be robust to difference between an offline training dataset and the real world (i.e., it must generalize), and must be robust to non-stationary changes in the real world itself (i.e., it must ignore visual distractors), and it must be equipped with mechanisms to actively explore to gather information about connectivity and traversability. Different environments typically exhibit similar physical structures, and these similarities can be used to accelerate exploration of *new* environments. Learning-based methods provide an appealing approach for learning a representation of this shared structure using prior experience.

In this work, we consider the problem of navigating to a user-specified goal in a previously unseen environment. The robot has access to a large and diverse dataset of experience from *other* environments, which it can use to learn general navigational affordances. Our approach to this problem uses a an information bottleneck architecture to learn a compact representation of goals. Learned from prior data, this latent goal model encodes prior knowledge about perception, navigational affordances, and short-horizon control. We use a non-parametric memory to incorporate experience from the new environment. Combined, these components enable our system to learn to navigate to goals in a new environment after only a few minutes of exploration.

The primary contribution of this work is a method for exploring novel environments to discover user-specified goals. Our method operates directly on a stream of image observations, without relying on structured sensors or geometric maps. An important part of our method is a compressed representation of goal images that simultaneously affords robustness while providing a simple mechanism for exploration. Such a representation allows us, for example, to specify a goal image at one time of day, and then navigate to that same place at a different time of day: despite variation in appearance, the latent goal representations must be sufficiently close that the model can produce the correct actions. Robustness of this kind is critical in real-world settings, where the appearance of landmarks can change significantly with times of day and seasons of the year.

Figure 1: We demonstrate RECON on a Clearpath Jackal.

We demonstrate our method, **R**apid **E**xploration **C**ontrollers for **O**utcome-driven **N**avigation (RECON), on a mobile ground robot (Fig. 1) and rigorously evaluate against 6 competitive baselines

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

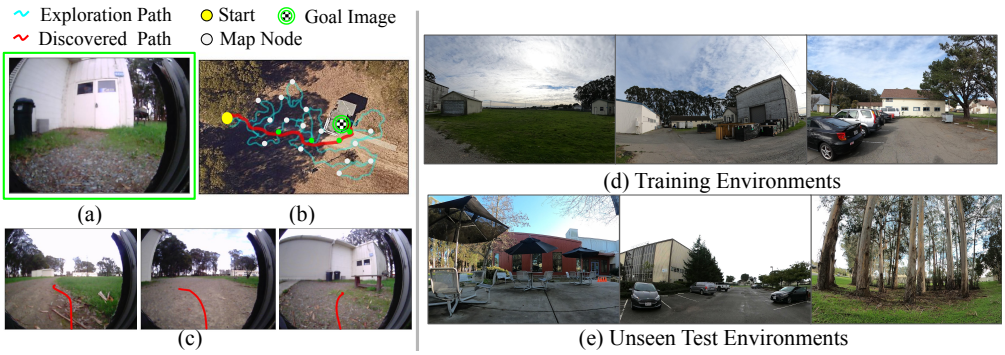


Figure 2: System overview: Given a goal image *(a)*, RECON explores the environment *(b)* by sampling prospective *latent* goals and constructing a topological map of images (white dots), operating only on visual observations. After finding the goal *(c)*, RECON can reuse the map to reach arbitrary goals in the environment (red path in *(b)*). RECON uses data collected from diverse training environments *(d)* to learn navigational priors that enable it to quickly explore and learn to reach visual goals a variety of unseen environments *(e)*.

spanning over 100 hours of real-world experiments in 8 distinct open-world environments (Fig. 2). Our method can discover user-specified goals up to 80m away after just 20 minutes of interaction in a new environment. We also demonstrate robustness in the presence of visual distractors and novel obstacles.

2 Related Work

Exploring a new environment is often framed as the problem of efficient mapping, posed in terms of information maximization to guide the robot to uncertain regions of the environment. Some prior exploration methods use local strategies for generating control actions for the robots [1–4], while others use use global strategies based on the frontier method [5–7]. However, building high-fidelity geometric maps can be hard without reliable depth information. Further, such maps do not encode semantic aspects of traversability, e.g., tall grass is traversable but a wire fence is not.

Inspired by prior work [8–12], we construct a topological map by learning a distance function and a low-level policy. We estimate distances via supervised regression and learn a local control policy via goal-conditioned behavior cloning [13, 14]. However, these prior methods do not describe how to learn to navigate in *new*, unseen environments. We equip RECON with an explicit mechanism for exploring new environments and transferring knowledge across environments.

Well-studied methods for exploration in reinforcement learning (RL) utilize a novelty-based bonus, computed from a predictive model [15–21], information gain [22, 23], or methods based on counts, densities, or distance from previously-visited states [24–26]. However, these methods learn to reason about the novelty of a state only after visiting it. Recent works [27, 28] improve upon this by predicting explorable areas for interesting parts of the environment to accelerate visual exploration. While these methods can yield state-of-the-art results in simulated domains [29, 30], they come at the cost of high sample complexity (over 1M samples) and are infeasible to train in open-world environments without a simulated counterpart. Instead, our method enables the robot to explore an environment from scratch in just 20 minutes, using prior experience from other environments.

The problem of reusing experience across tasks is studied in the context of meta-learning [31–33] and transfer learning [34–38]. Our method uses an information bottleneck [39], which serves a dual purpose: first, it provides a representation that can aid the generalization capabilities of RL algorithms [40, 41], and second, it serves as a measure of task-relevant uncertainty [42], allowing us to incorporate prior information for proposing goals that are functionally-relevant for learning control policies in the new environment.

The problem of learning goal-directed behavior has been studied extensively using RL [43–46] and imitation learning (IL) [13, 14, 47–50]. Our method builds upon prior goal-conditioned IL methods to solve a slightly different problem: how to reach goals in a *new* environment. Once placed in a new environment, our method explores by carefully choosing which goals to visit, inspired by prior work [51–55]. Unlike these prior methods, however, our method makes use of previous experience in *different* environments to accelerate learning in the current environment.

3 Problem Statement and System Overview

We consider the problem of goal-directed exploration for visual navigation in novel environments: a robot is tasked with navigating to a goal location G, given an image observation o_g taken at G. Broadly, this consists of three separate stages: (1) learning from offline data, (2) building a map in a new environment, and (3) navigating to goals in the new environment. We model the task of navigation as a Markov decision process with time-indexed states $s_t \in \mathcal{S}$ and actions $a_t \in \mathcal{A}$. We *do not assume* the robot has access to spatial localization or a map of the environment, or access to the system dynamics. We use videos of robot trajectories in a variety of environments to learn general navigational skills and build a compressed representation of the perceptual inputs, which can be used to guide the exploration of novel environments. We make no assumption on the nature of the trajectories: they may be obtained by human teleoperation, self-exploration, or as a result of a preset policy. These trajectories need not exhibit intelligent behavior. Since the robot only observes the world from a single on-board camera and does not run any state estimation, our system operates in a partially observed setting. Our system commands continuous linear and angular velocities.

3.1 Mobile Robot Platform

We implement RECON on a Clearpath Jackal UGV platform (see Fig. 1). The default sensor suite consists of a 6-DoF IMU, a GPS unit for approximate global position estimates, and wheel encoders to estimate local odometry. In addition, we added a forward-facing 170° field-of-view RGB camera and an RPLIDAR 2D laser scanner. Inside the Jackal is an NVIDIA Jetson TX2 computer. The GPS and laser scanner can become unreliable in some environments [56], so we use them solely as safety controllers during data collection. Our method operates only using images taken from the onboard RGB camera, without other sensors or ground-truth localization.

3.2 Self-Supervised Data Collection & Labeling

Our aim is to leverage data collected in a wide range of different environments to enable the robot to discover and learn to navigate to novel goals in novel environments. We curate a dataset of self-supervised trajectories collected by a time-correlated random walk in diverse real-world environments (see Fig. 2 (d,e)). This data was collected over a span of 18 months and exhibits significant variation in appearance due to seasonal and lighting changes. We make this dataset publicly available and provide further details in Appendix A.

4 RECON : A Method for Goal-Directed Exploration

Our objective is to design a robotic system that uses visual observations to efficiently discover and reliably reach a target image in a previously unseen environment. RECON consists of two components that enable it to explore new environments. The first is an uncertainty-aware, context-conditioned representation of goals that can quickly adapt to novel scenes. The second component is a topological map, where nodes represent egocentric observations and edges are the predicted distance between them, constructed incrementally from frontier-based exploration, maintaining a compact memory of the target environment.

4.1 Learning to Represent Goals

Our method learns a compact representation of goal images that is robust to task-irrelevant factors of variation. We learn this representation using a variant of the information bottleneck architecture [42, 57]. We use a context-conditioned representation of goals to learn a control policy in the target environment (Fig. 3 describes the graphical model). Letting $I(\cdot;\cdot)$ denote mutual information, the objective in Eq. 1 encourages the model to compress the incoming goal image o_g into a representation z_t^g conditioned on the current observation o_t that is predictive of the best action a_t^g and the temporal distance d_t^g to the goal (upper-case denotes random variables):

$$I\big((A_t^g, D_t^g); Z_t^g \mid o_t\big) - \beta I(Z_t^g; O_g \mid o_t) \tag{1}$$

Following [42], we approximate the intractable objective in Eq. 1 with a variational posterior and decoder (an upper bound), resulting in the maximization objective:

$$L = \frac{1}{|\mathcal{D}|} \sum_{(o_t, o_g, a_t^g, d_t^g) \in \mathcal{D}} \mathbb{E}_{p_\phi(z_t^g | o_g, o_t)} \left[\log q_\theta \left(a_t^g, d_t^g \mid z_t^g, o_t \right) \right] - \beta \mathrm{KL} \left(p_\phi(\cdot \mid o_g, o_t) || r(\cdot) \right) \quad (2)$$

where we define the prior $r(z_t^g) \triangleq \mathcal{N}(0, I)$ and $\mathcal{D}$ is a dataset of trajectories characterized by (o_t, o_g, a_t^g, d_t^g) quadruples. The first term measures the model's ability to predict actions and distances from the encoded representation, and the second term measures the model's compression of incoming goal images.

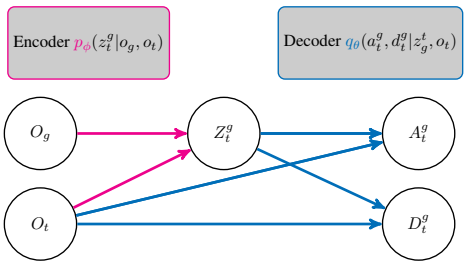


Figure 3: Graphical model of actions and distances

As the encoder p_ϕ and decoder q_θ are conditioned on o_t, the representation z_t^g only encodes information about *relative* location of the goal from the context – this allows the model to represent *feasible* goals. If, instead, we had a typical VAE (in which the input images are autoencoded), the samples from the prior over these representations would not necessarily represent goals that are reachable from the current state. This distinction is crucial when exploring *new* environments, where most states from the training environments are not valid goals.

4.2 Goal-Directed Exploration with Topological Memory

The second component of our system is a topological memory constructed incrementally as the robot explores a new environment. It provides an estimate of the exploration frontier as well as a map that the robot can use to later navigate to arbitrary goals. To build this memory, the robot uses the model from the previous section to propose *subgoals* for data collection. Note that this is done in the exploration phase and have a latent goal model pre-trained on the offline dataset. Given a subgoal, our algorithm (Alg. 1) proceeds by executing actions towards the subgoal for a fixed number of timesteps (Alg. 1 L12). The data collected during subgoal navigation expands the topological memory (Alg. 1 L14) and is used to fine-tune the model (Alg. 1 L15). Thus, the task of efficient exploration is reduced to the task of choosing subgoals.

Subgoals are represented by latent variables in our model, which may either come from the posterior $p_\phi(z|o_t, o_g)$, or from the prior $r(z)$. Given a subgoal z and observation o_t, the model decodes it into

Algorithm 1 *RECON for Exploration:* RECON takes as input an encoder p_ϕ, a decoder q_θ, prior r, the current observation o_t and goal observation o_g. $\delta_1, \delta_2, \epsilon, \beta \in \mathbb{R}_+$; $H, \gamma \in \mathbb{N}$ are hyperparameters.

1: **function** RECON $(q_\theta, p_\phi, r, o_t, o_g; \delta_1, \delta_2, \epsilon, \beta, \gamma, H)$
2: $\quad \mathcal{G} \leftarrow \emptyset, \mathcal{D} \leftarrow \emptyset$ ▷ *Initialize graph and data*
3: $\quad$ **while** not reached goal $[\bar{d}_t^g < \delta_1]$ **do** ▷ *Continue while not at goal*
4: $\quad\quad o_n \leftarrow \mathrm{LeastExploredNeighbor}(\mathcal{G}, o_t; \delta_2)$
5: $\quad\quad z_t^g \sim p_\phi(z \mid o_t, o_g)$ ▷ *Encode relative goal*
6: $\quad\quad$ **if** goal is feasible $[r(z_t^g) > \epsilon]$ **then**
7: $\quad\quad\quad z_t^w \leftarrow z_t^g$ ▷ *Will go to the goal*
8: $\quad\quad$ **else if** robot at frontier $[\bar{d}_t^n < \delta_1]$ **then**
9: $\quad\quad\quad z_t^w \sim r(z)$ ▷ *Will explore from frontier*
10: $\quad\quad$ **else**
11: $\quad\quad\quad z_t^w \sim p_\phi(z \mid o_t, o_n)$ ▷ *Will go to frontier*
12: $\quad\quad \mathcal{D}_w, o_t \leftarrow \mathrm{SubgoalNavigate}(z_t^w; H)$
13: $\quad\quad \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_w$
14: $\quad\quad \mathrm{ExpandGraph}(\mathcal{G}, o_t)$
15: $\quad\quad$ Step $L(\phi, \theta; \mathcal{D}, \beta)$ for γ epochs ▷ Eq. 2
16: $\quad$ **return** networks p_ϕ, q_θ and graph $\mathcal{G}$

an action and distance pair $q(a_t^g, d_t^g | z, o_t)$; the action is used to control the robot towards the goal, and the distance is used to construct edges in the topological graph. The choice of intermediate subgoal to navigate toward at any step is based on the robot's estimate of the goal reachability and its proximity to the frontier. To determine the frontier of the graph, we track the number of times each node in the graph was selected as the navigation goal; nodes with low counts are considered to be on the frontier. In the following, we use $\bar{z}_t^g$ to denote the mean of the encoder $p_\phi(z \mid o_t, o_g)$, and $\bar{d}_t^g$ to denote the distance component of the mean of the decoder $q_\theta(a_t, d_t \mid \bar{z}_t^g, o_t)$ (i.e., the predicted number of time steps from o_t to $\bar{z}_t^g$). The choice of subgoal at each step is made as follows:

(i) Feasible Goal: The robot believes it can reach the goal and adopts the representation of the goal image as the subgoal (Alg. 1 L7). The robot's confidence in reaching the goal is based on the probability of the current goal embedding z_t^g under the prior $r(z)$. Large $r(z_t^g)$ implies the relationship between the observation o_t and the goal o_g is *in-distribution*, suggesting that the model's estimates of the distances is reliable – intuitively, this means that the model is confident about the distance to o_g and can reach it.

Algorithm 2 *RECON for Goal-Reaching:* After exploration, RECON uses the topological graph $\mathcal{G}$ to quickly navigate towards the goal o_g.

1: **procedure** GoalNavigate($\mathcal{G}, o_t, o_g; H$)
2: $\quad v_t \leftarrow$ AssociateToVertex($\mathcal{G}, o_t$)
3: $\quad v_g \leftarrow$ AssociateToVertex($\mathcal{G}, o_g$)
4: $\quad (v_t, \ldots, v_g) \leftarrow$ ShortestPath($\mathcal{G}, v_t, v_g$)
5: $\quad$ **for** $v \in (v_t, \ldots, v_g)$ **do**
6: $\quad\quad z \leftarrow p_\phi(z \mid o_t, o_g = v.\text{o})$
7: $\quad\quad \mathcal{D}_w, o_t \leftarrow$ SubgoalNavigate($z; H$)

(ii) Explore at Frontier: The robot is at the "least-explored node" (frontier) o_n and explores by sampling a random conditional subgoal latent z_t^w from the prior (Alg. 1 L9). The robot determines whether it is at the frontier based on the distance (estimated by querying the model) to its "least explored neighbor" $\bar{d}_t^n$ – the node in the graph within a distance threshold (δ_2) of the current observation that has the lowest visitation count. If the distance to this node $\bar{d}_t^n$ is low (threshold δ_1), then the robot is at the frontier.

(iii) Go to Frontier: The robot adopts its "least-explored neighbor" o_n as a subgoal (Alg. 1 L11).

The SubgoalNavigate function rolls out the learned policy for a fixed time horizon H to navigate to the desired subgoal latent z_t^w, by querying the decoder $q_\theta(a_t, d_t | z_t^w, o_\tau)$ with a fixed subgoal latent. The endpoint of such a rollout is used to update the visitation counts in the graph $\mathcal{G}$. At the end of each trajectory, the ExpandGraph subroutine is used to update the edge and node sets $\{\mathcal{E}, \mathcal{V}\}$ of the graph $\mathcal{G}$ to update the representation of the environment. We provide the pseudocode for these subroutines in Appendix B.1. We also share broader implementation details including choice of hyperparameters, model architectures and training details in Appendix B.2.

4.3 System Summary

RECON uses the latent goal model and topological graph to quickly explore new environments and discovers user-specified goals. Our complete system consists of three stages:

A) *Prior Experience:* The goal-conditioned distance and action model (Sec. 4.1) is trained using experience from previously visited environments. Supervision for training our model is obtained by using time steps as a proxy for distances and a relabeling scheme (Appendix A).

B) *Exploring a Novel Environment:* When placed in a new environment, RECON uses a combination of frontier-based exploration and latent goal-sampling with the learned model. The learned model is also fine-tuned to this environment. These steps are summarized in Alg. 1 and Sec. 4.2.

C) *Navigating an Explored Environment:* Given an explored environment (represented by a topological graph $\mathcal{G}$) and the model, RECON uses $\mathcal{G}$ to navigate to a goal image by planning a path of subgoals through the graph. This process is summarized in Alg. 2.

5 Experimental Evaluation

We designed our experiments to answer four questions:

Q1. How does RECON compare to prior work for visual goal discovery in novel environments?

Method	Expl. Time (mm:ss) ↓	Nav. Time (mm:ss) ↓	SCT [58] ↑
PPO + RND [18]	21:18	00:47	0.22
InfoBot [41]	23:36	00:48	0.21
Active Neural SLAM (ANS) [21]	21:00	00:45	0.33
ViNG [11]	19:48	00:34	0.60
Ours + Episodic Curiosity (ECR) [20]	14:54	00:31	0.73
RECON (Ours)	**09:54**	**00:26**	**0.92**

Table 1: Exploration and goal reaching performance: Exploring 8 real-world environments, RECON reaches the goal 50% faster than the best baseline (ECR). ANS takes up to 2x longer to find the goal and NTS [27] fails to find the goal in every environment. On subsequent traversals, RECON navigates to the goal 20–85% faster than other baselines, and exhibits $>30\%$ higher weighted success.

Q2. After exploration, can RECON leverage its experience to navigate to the goal efficiently?

Q3. What is the range of perturbations and non-stationary elements to which RECON is robust?

Q4. How important are the various components of RECON, such as sampling from an information bottleneck and non-parametric memory, to its performance?

5.1 Goal-Directed Exploration in Novel Environments

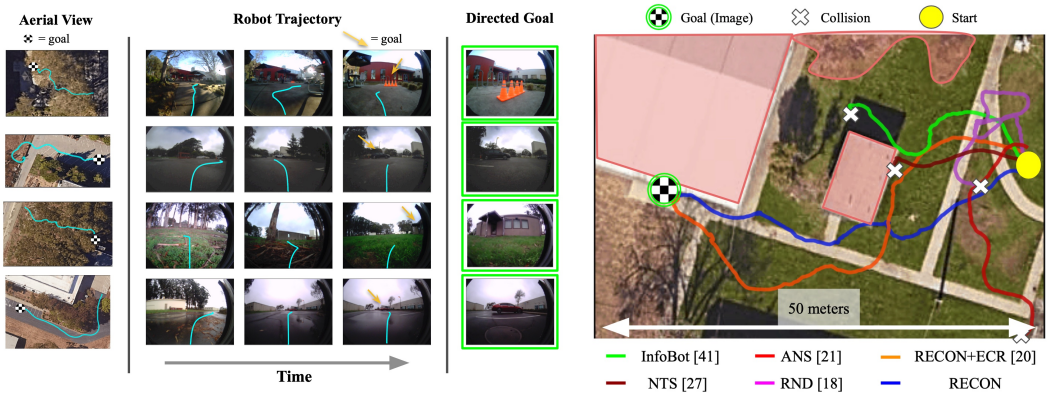


Figure 4: Visualizing goal-reaching behavior of the system: *(left)* Example trajectories to goals discovered by RECON in *previously unseen* environments. *(right)* Policies learned by the different methods in one such environment. Only RECON and ECR reach the goal successfully, and RECON takes the shorter route.

We perform our evaluation in a diverse variety of outdoor environments (examples in Fig. 2), including parking lots, suburban housing, sidewalks, and cafeterias. We train our self-supervised navigation model using an offline navigation dataset (Sec.3.2) collected in a distinct set of training environments, and evaluate our system's ability to discover user-specified goals in previously unseen environments. We compare RECON to six baselines, each trained on the same 20 hours of offline data as our method, and finetuned in the target environment with online interaction.

1. **PPO + RND:** Random Network Distillation (RND) is a widely used prediction bonus-based exploration strategy in RL [18], which we use with PPO [59, 60]. This comparison is representative of a frequently used approach for exploration in RL using a novelty-based bonus.
2. **InfoBot:** An offline variant of InfoBot [41], which uses goal-conditioned information bottleneck, analogous to our method, but does not use the non-parametric memory.
3. **Active Neural SLAM (ANS):** A popular indoor navigation approach based on metric spatial maps proposed for coverage-maximizing exploration [21]. We adapt it to the goal-directed task by using the distance function from RECON to detect when the goal is nearby.
4. **Neural Topological SLAM (NTS):** A recent approach that uses image-space predictions of semantically rich areas to explore, with topological maps serving as an episodic memory [27]. This baseline represents the state-of-the-art in visual navigation for structured indoor environments that use high-fidelity simulators, such as Habitat or AI2-THOR [29, 30].

5. **Visual Navigation with Goals (ViNG):** A method that uses random action sequences to explore and incrementally build a topological graph without reasoning about visitation counts [11].
6. **Episodic Curiosity (ECR):** A method that executes random action sequences at the frontier of a topological graph for exploration [20]. We implement this as an *ablation* of our method that samples random action sequences, rather than rollouts to sampled goals (Alg. 1 Line 7).

We evaluate the ability of RECON to discover visually-indicated goals in 8 *unseen* environments and navigate to them repeatedly. For each trial, we provide an RGB image of the desired target (one per environment) to the robot and report the time taken by each method to *(i)* discover the desired goal (**Q1**), and *(ii)* reliably navigate to the discovered goal a second time using prior exploration (**Q2**). Additionally, we quantify navigation performance using Success weighed by Completion Time (SCT), a success metric that takes into account the agent's dynamics [58]. We show quantitative results in Table 1, and visualize sample trajectories of RECON and the baselines in Fig. 4.

RECON outperforms all the baselines, discovering goals that are up to 80m away in under 20 minutes, including instances where no other baseline can reach the goal successfully. RECON+ECR and ViNG discover the goal in only the easier environments, and take up to 80% more time to discover the goal in those environments. RND, InfoBot and ANS are able to discover goals that are up to 25m away but fails to discover more distant goals, likely because using reinforcement learning for fine-tuning is data-inefficient. We exclude reporting metrics on NTS, which fails to successfully explore any environment, likely due to overfitting to the offline trajectories. Indeed, the simulation experiments reported in each of these online algorithms require upwards of 1M timesteps to adapt to new environments [21, 27, 41]. We attribute RECON's success to the context-conditioned sampling strategy (described in Sec. 4.1), which proposes goals that can accelerate the exploration of new environments.

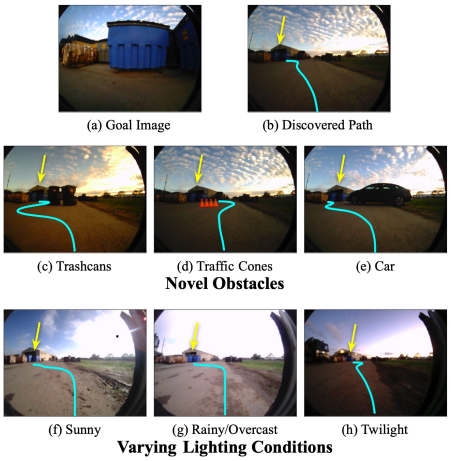


Figure 5: Exploring non-stationary environments: The learned representation and topological graph is robust to visual distractors, enabling reliable navigation to the goal under novel obstacles *(c–e)* and appearance changes *(f–h)*.

We then study RECON's ability to quickly reach goals after initial discovery. Table 1 shows that RECON variants are able to quickly recall a feasible path to the goal. These methods create a compact topological map from experience in the target environment, allowing them to quickly reach previously-seen states. The other baselines are unsuccessful at recalling previously seen goals for all but the simplest environments. Fig. 4 shows an aerial view of the paths recalled by various methods in one of the environments. Only the RECON variants are successfully able to navigate to the checkerboard goal; all other baselines result in collisions in the environment. Further, RECON discovers a shorter path to the goal and takes 30% less time to navigate to it than ECR ablation.

5.2 Exploring Non-Stationary Environments

Outdoor environments exhibit non-stationarity due to dynamic obstacles, such as automobiles and people, as well as changes in appearance due to seasons and time of day. Successful exploration and navigation in such environments requires learning a representation that is invariant to such distractors. This capability is of central interest when using a non-parametric memory: for the topological map to remain valid when such distractors are presented, we must ensure the invariance of the learned representation to such factors (**Q3**).

To test the robustness of RECONto unseen obstacles and appearance changes, we first had RECON explore in a new "junkyard" to learn to reach a goal image containing a blue dumpster (Fig. 5-a). Then, without any more exploration, we evaluated the learned goal-reaching policy when presented with *previously unseen* obstacles (trash cans, traffic cones, and a car) and and weather conditions (sunny, overcast, and twilight). Fig. 5 shows trajectories taken by the robot as it successfully navigates to the goal in scenarios with varying obstacles and lighting conditions. These results

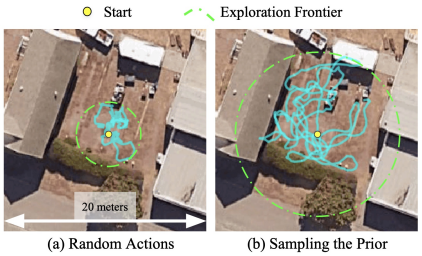


Figure 6: Exploration via sampling from our context-conditioned prior (*right*) allows the robot to explore 5 times faster than using random actions, e.g. in ECR [20] (*left*).

Method	Expl. Time (mm:ss) ↓	Nav. Time (mm:ss) ↓	SCT [58] ↑
Reactive	11:54	00:37.4	0.63
Random Actions	14:54	00:31.4	0.73
Vanilla Sampling	14:06	00:28.7	0.83
Ours	**09:56**	**00:25.8**	**0.92**

Table 2: Ablation experiments confirm the importance of using an information bottleneck and a non-parametric memory.

suggest that the learned representations are invariant to visual distractors that do not affect robot's decisions to reach a goal (e.g., changes in lighting conditions do not affect the trajectory to goal, and hence, are discarded by the bottleneck).

5.3 Dissecting RECON

RECON explores by sampling goals from the prior distribution over state-goal representations. To quantify the importance of this exploration strategy (**Q4**), we deploy RECON to perform undirected exploration in a novel target environment *without building a graph* of the environment. We compare the coverage of trajectories of the robot over 5 minutes of exploration when: *(a)* it executes random action sequences [20], and *(b)* it performs rollouts towards sampled goals. We see that performing rollouts to sampled goals results in 5x faster exploration in novel environments (see Fig. 6).

We also evaluate several variants of RECON that ablate its goal sampling and non-parametric memory on the end-to-end task of visual goal discovery in novel environments:

- *Reactive:* our method deployed *without* the topological graph for memory.
- *Random Actions:* a variant of our method that executes random action sequences at the frontier rather than rollouts to sampled goals. This is identical to the ECR baseline described in Sec. 5.1.
- *Vanilla Sampling:* a variant of our method which learns a goal-conditioned policy and distances *without* an information bottleneck to obtain compressed representations.

We deploy these variants in a subset of the unseen test environments and summarize their performance in Table 2. These results corroborate the observations in Fig. 6: learning a compressed goal representation is key to the performance of RECON. "Vanilla Sampling", despite sampling from a joint prior, performs poorly and is unable to discover distant goals. We hypothesize that our method is more robust because the information bottleneck helps learn a representation that ignores task-irrelevant information. We also observe that "Reactive" experiences a smaller degradation in exploration performance, suggesting that goal-sampling can help with the exploration problem even without the graph. However, we find a massive degradation in its ability to recall previously discovered goals, suggesting that the memory is key to the navigation performance of RECON.

6 Discussion

We proposed a system for efficiently learning goal-directed policies in new open-world environments. The key idea behind our method is to use a learned goal-conditioned distance model with a latent variable model representing visual goals for rapid goal-directed exploration. The problem setup studied in this paper, using past experience to accelerate learning in a *new* environment, is reflective of real-world robotics scenarios: collecting new experience at deployment time is costly, but experience from prior environments can provide useful guidance to solve new tasks.

In future work, we aim to provide theoretical guarantees for when and where we can expect stochastic policies and the information bottleneck to provide efficient exploration. One limitation of the current method is that it does not explicitly account for the value of information. Accounting for such states can generate a better goal-reaching policy.

References

[1] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and Autonomous Systems*, 1991, special Issue Toward Learning Robots.

[2] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2002, pp. 540–545 vol.1.

[3] T. Kollar and N. Roy, "Efficient optimization of information-theoretic exploration in slam," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

[4] W. Tabib, K. Goel, J. Yao, M. Dabhi, C. Boirum, and N. Michael, "Real-time information-theoretic exploration with gaussian mixture model maps." in *Robotics: Science and Systems*, 2019.

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1997.

[6] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using cauchy-schwarz quadratic mutual information," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[7] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "A comparative evaluation of exploration strategies and heuristics to improve them," 2011.

[8] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-Parametric Topological Memory for Navigation," in *International Conference on Learning Representations*, 2018.

[9] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5113–5120.

[10] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the Replay Buffer: Bridging Planning and RL," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[11] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "ViNG: Learning Open-World Navigation with Visual Goals," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[12] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling Local Control to Large-Scale Topological Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[13] D. Ghosh, A. Gupta, J. Fu, A. Reddy, C. Devin, B. Eysenbach, and S. Levine, "Learning to reach goals without reinforcement learning," *arXiv preprint arXiv:1912.06088*, 2019.

[14] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on Robot Learning*. PMLR, 2020, pp. 1113–1132.

[15] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015.

[16] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2778–2787.

[17] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[18] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[19] Y. Satsangi, S. Lim, S. Whiteson, F. Oliehoek, and M. White, "Maximizing information gain in partially observable environments via prediction reward," *arXiv preprint arXiv:2005.04912*, 2020.

[20] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," *International Conference on Learning Representations (ICLR*, 2019.

[21] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to Explore using Active Neural SLAM," in *International Conference on Learning Representations (ICLR)*, 2020.

[22] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," *arXiv preprint arXiv:1605.09674*, 2016.

[23] A. Mirchev, B. Kayalibay, M. Soelch, P. van der Smagt, and J. Bayer, "Approximate bayesian inference in spatial environments," *arXiv preprint arXiv:1805.07206*, 2018.

[24] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," *arXiv preprint arXiv:1606.01868*, 2016.

[25] E. Hazan, S. Kakade, K. Singh, and A. Van Soest, "Provably efficient maximum entropy exploration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2681–2691.

[26] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov, "Efficient exploration via state marginal matching," *arXiv preprint arXiv:1906.05274*, 2019.

[27] D. Singh Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta, "Semantic curiosity for active visual learning," in *ECCV*, 2020.

[29] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[30] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *ArXiv*, vol. abs/1712.05474, 2019.

[31] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[32] S. Sæmundsson, K. Hofmann, and M. P. Deisenroth, "Meta reinforcement learning with latent variable gaussian processes," *arXiv preprint arXiv:1803.07551*, 2018.

[33] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[34] M. E. Taylor and P. Stone, "Cross-domain transfer for reinforcement learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 879–886.

[35] A. Lazaric, "Transfer in reinforcement learning: a framework and a survey," in *Reinforcement Learning*. Springer, 2012, pp. 143–173.

[36] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.

[37] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *arXiv preprint arXiv:1703.02949*, 2017.

[38] S. Gamrian and Y. Goldberg, "Transfer learning for related reinforcement learning tasks via image-to-image translation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2063–2072.

[39] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[40] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, and K. Hofmann, "Generalization in reinforcement learning with selective noise injection and information bottleneck," *arXiv preprint arXiv:1910.12911*, 2019.

[41] A. Goyal, R. Islam, D. Strouse, Z. Ahmed, M. Botvinick, H. Larochelle, Y. Bengio, and S. Levine, "Infobot: Transfer and exploration via the information bottleneck," *arXiv preprint arXiv:1901.10902*, 2019.

[42] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[43] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*. Citeseer, 1993, pp. 1094–1099.

[44] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal Value Function Approximators," in *International Conference on Machine Learning (ICML)*, 2015.

[45] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep rl for model-based control," *arXiv preprint arXiv:1802.09081*, 2018.

[46] B. Eysenbach, R. Salakhutdinov, and S. Levine, "C-learning: Learning to achieve goals via recursive classification," *arXiv preprint arXiv:2011.08909*, 2020.

[47] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel, "Goal-conditioned imitation learning," *arXiv preprint arXiv:1906.05838*, 2019.

[48] H. Sun, Z. Li, X. Liu, D. Lin, and B. Zhou, "Policy continuation with hindsight inverse dynamics," *arXiv preprint arXiv:1910.14055*, 2019.

[49] R. K. Srivastava, P. Shyam, F. Mutz, W. Jaśkowski, and J. Schmidhuber, "Training agents using upside-down reinforcement learning," *arXiv preprint arXiv:1912.02877*, 2019.

[50] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," in *International Conference on Learning Representations*, 2020.

[51] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine, "Skew-fit: State-covering self-supervised reinforcement learning," *arXiv preprint arXiv:1903.03698*, 2019.

[52] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, and P.-Y. Oudeyer, "Curious: intrinsically motivated modular multi-goal reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 1331–1340.

[53] L. Zhang, G. Yang, and B. C. Stadie, "World model as a graph: Learning latent landmarks for planning," *arXiv preprint arXiv:2011.12491*, 2020.

[54] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba, "Maximum entropy gain exploration for long horizon multi-goal reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7750–7761.

[55] K. Liu, T. Kurutach, C. Tung, P. Abbeel, and A. Tamar, "Hallucinative topological memory for zero-shot visual planning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6259–6270.

[56] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An Autonomous Self-Supervised Learning-Based Navigation System," 2020.

[57] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 1947–1980, 2018.

[58] N. Yokoyama, S. Ha, and D. Batra, "Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation," 2021.

[59] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017.

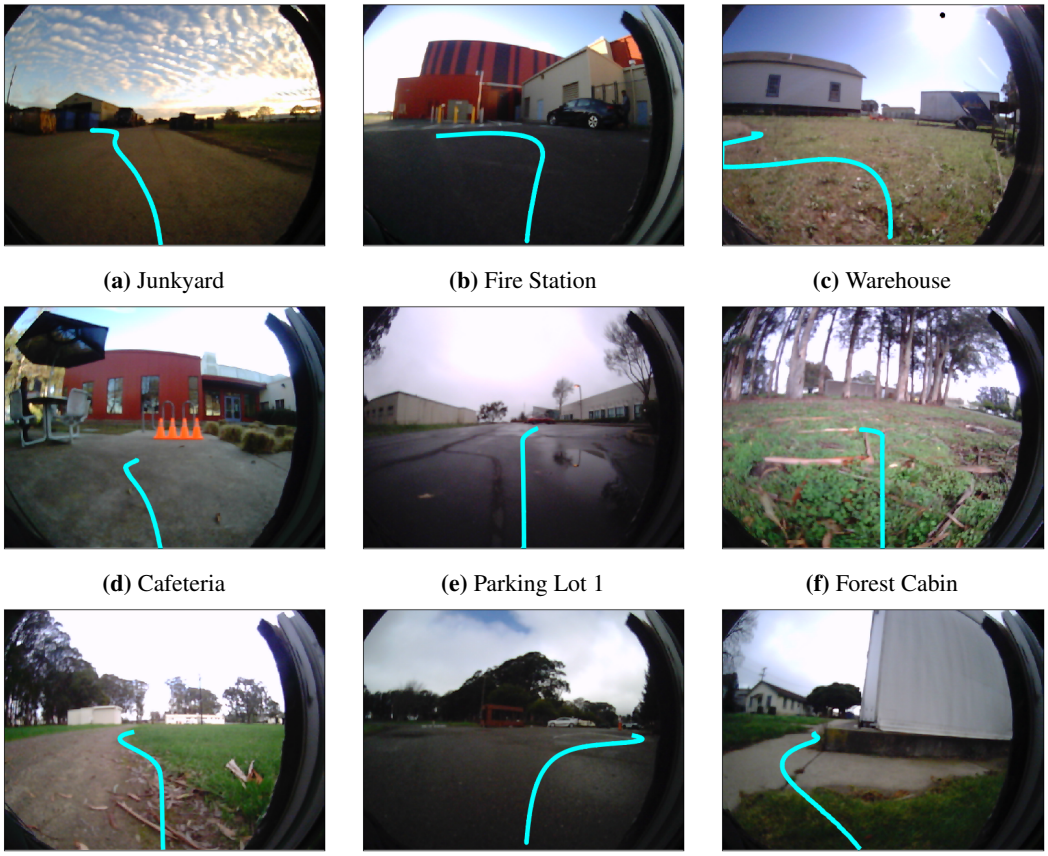
[60] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," in *International Conference on Learning Representations (ICLR)*, 2020.

[61] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[62] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

A Dataset

In this work, we emphasize that data collected from prior experience in unrelated environments can be a rich source of supervision, even if the interactions in the dataset are suboptimal. To demonstrate this, we curate a dataset of over 5000 self-supervised trajectories collected over 9 distinct real-world environments. These trajectories capture the interaction of the robot in diverse environments, including phenomena like collisions with obstacles and walls, getting stuck in the mud or pits, or flipping due to bumpy terrain. The dataset contains measurements from a wide range of sensors including a pair of stereo RGB cameras, thermal camera, 2D LiDAR, GPS and IMU to support offline evaluation using an alternative suite of sensors. While a lot of these sensor measurements can be noisy and unreliable, we believe that learning-based techniques coupled with multimodal sensor fusion can provide a lot of benefits in the real-world. This dataset was collected over a span of 18 months and exhibits significant variation in appearance due to seasonal and lighting changes[1].



(a) Junkyard **(b)** Fire Station **(c)** Warehouse

(d) Cafeteria **(e)** Parking Lot 1 **(f)** Forest Cabin

(g) Farmlands **(h)** Parking Lot 2 **(i)** Residential

Figure 7: We collect data in 9 diverse environments. Example trajectories are shown in cyan.

A.1 Self-Supervised Data Collection and Labeling

We design the data collection methodology to enable gathering large amounts of diverse data with minimal human intervention. Due to the high cost of gathering data with real-world robotic systems, we choose to use an off-policy learning algorithm in order to be able to gather data using any control policy and train on all of the gathered data. To ensure that the control policy achieves sufficient

[1] A sizable portion of the dataset also contains data collected by our collaborators for other projects. The ability to use data collected by other researchers without the same intent is a major strength of our method. We retract references to this work to respect anonymity policies of the peer-review process and will share more details in the public release.

coverage of the environment while also ensuring that the action sequences executed by the robot are realistic, we use a time-correlated random walk to gather data. A naïve uniform random control policy is inadequate because the robot will primarily drive straight due to the linear and angular velocity action interface of the robot, which will result in both insufficient exploration and unrealistic test time action sequences.

During data collection using the random control policy, the robot requires a mechanism to detect if it is in collision or stuck, and an automated controller to reset itself in order to continue gathering data. We detect collisions in one of two ways, either using the LIDAR to detect when an obstacle is near or the IMU to detect when the robot is stuck due to an obstacle or uneven terrain. We program an automated backup maneuver that drives the robot out of collision (whenever possible) so it can initiate a new trajectory.

We also use these collision detectors as a weak source of supervision by generating *event labels* for the collected trajectories, giving us a self-supervised relabeling pipeline as proposed in BADGR [56]. We consider three different events: collision, bumpiness, and position. A collision event is detected the LIDAR measures an obstacle to be close or, in off-road environments, when the IMU detects a sudden drop in linear acceleration (jerk) and angular velocity magnitudes. A bumpiness event is calculated as occurring when the angular velocity magnitudes measured by the IMU are above a certain threshold. The position is determined by an onboard state estimator that fuses wheel odometry and the IMU to form a local position estimate. Note that all experiments reported in this paper only use the collision labels; these labels are used to dissect the random walks into smooth trajectories that end in collision.

A.2 Environments

To learn general navigational affordances across a wide range of environments, we curate over 40 hours of trajectories in 9 diverse open-world environments of varying complexity (see Figure 7).

Figure 8 shows the exploration and navigation performance of RECON and the baselines (see Sec. 5 for details) on the individual environments. As the environment complexity increases, most methods are not able to explore the environment efficiently to discover the goal. For videos of our system exploring these environments, please check out the supplemental video submission.

B Reproducibility

B.1 Algorithmic Components

The $\mathrm{SubgoalNavigate}$ function rolls out the learned policy for a fixed time horizon H to navigate to the desired subgoal latent z_t^w, by querying the decoder $q_\theta(a_t, d_t | z_t^w, o_\tau)$ in an open loop manner. The endpoint of such a rollout is used to update the visitation counts v in the graph $\mathcal{G}$ using the $\mathrm{AssociateToVertex}$ subroutine. To nudge the robot to the frontier, we use a heuristic $\mathrm{LeastExploredNeighbor}$ routine that uses the visitation counts of the neighbors to identify unexplored areas in the local neighborhood. At the end of each trajectory, the $\mathrm{ExpandGraph}$ subroutine is used to update the edge and node sets $\{\mathcal{E}, \mathcal{V}\}$ of the graph $\mathcal{G}$ to update the non-parametric representation of the environment. Pseudocode for these subroutines are given in Alg. 3.

B.2 Implementation Details

Inputs to the encoder p_ϕ are pairs of observations of the environment – current and goal – represented by a stack of two RGB images obtained from the onboard camera at a resolution of 160×120 pixels. p_ϕ is implemented by a MobileNet encoder [61] followed by a fully-connected layer projecting the 1024-dimensional latents to a stochastic, context-conditioned representa-

Hyperparam.	Value	Meaning
δ_1	4	Threshold of identification
δ_2	15	Threshold of neighbors
ϵ	10^{-2}	Exploration threshold on prior
β	1.0	Model complexity
γ	10	Epochs to finetune model
H	5 seconds	Horizon to navigate to subgoal

Table 3: Hyperparameters used in our experiments.

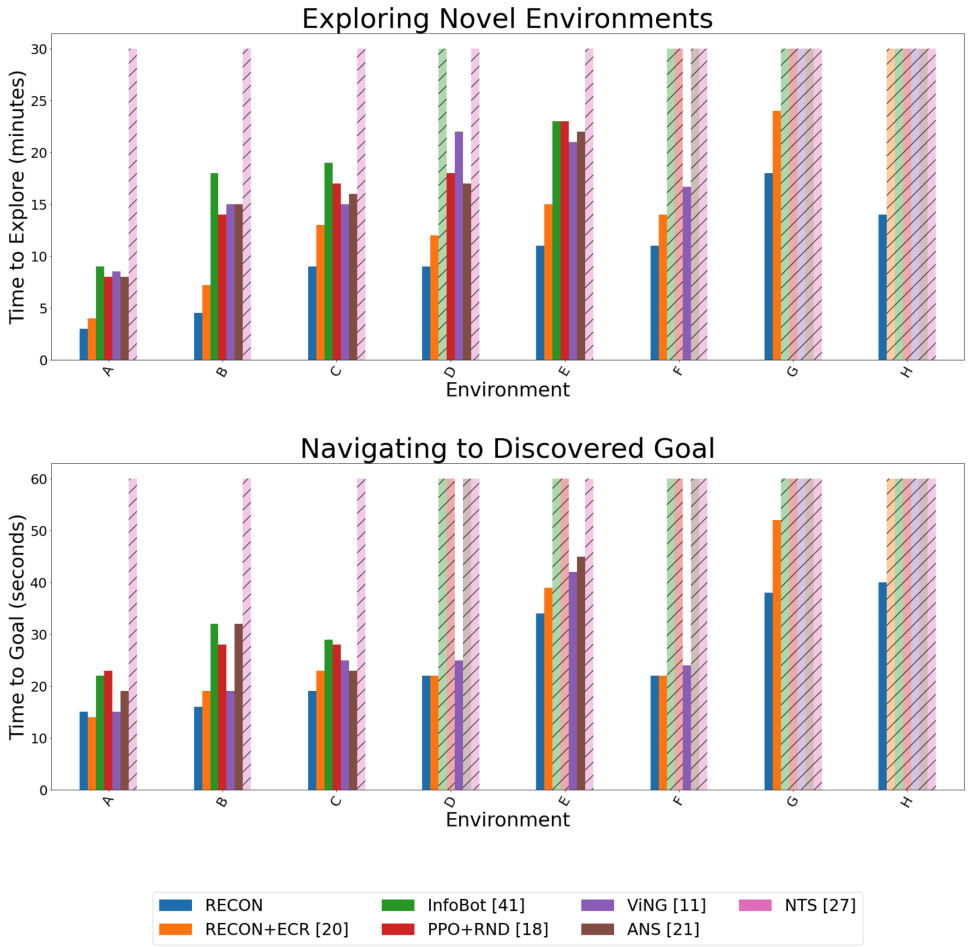


Figure 8: Exploring and learning to reach goals: *(left)* Amount of time needed for each method to search for the goals in a new environment (↓ is better; hashed out bars represent failure). *(right)* Amount of time needed to reach the goal a second time, after reaching the goal once and constructing the map, in seconds (↓ is better).

tion z_t^g of the goal that uses 64-dimensions each to represent the mean and diagonal covariance of a Gaussian distribution. Inputs to the decoder q_θ are the context (current observation) – processed with another MobileNet – and z_t^g. We use the reparametrization trick [62] to sample from the latent and use the concatenated encodings to learn the optimal actions a_t^g and distances d_t^g. Details of our network architecture are provided in Table 4. During pretraining, we maximize Eq. 2 with a batch size of 128 and perform gradient updates using the Adam optimizer with learning rate $\lambda = 10^{-4}$ until convergence. We provide the hyperparameters associated with our algorithms in Table 3.

C Supplemental Video

For more results and videos of our system deployed in unstructured, outdoor environments in the real-world, please check out the supplemental video submission.

Algorithm 3 Pseudocode for subroutines referenced in the exploration algorithm shown in Alg. 1

function SubgoalNavigate($z_t^w; H$)
  trajectory $\leftarrow ()$
  for $t \in [1, \ldots, H]$ **do**
    trajectory.append((o_t, a_t, t))
    $a_t, d_t^g \sim q_\theta(a_t, d_t \mid z_t^w, o_\tau)$ ▷ *Sample action*
    $o_t \leftarrow$ Env.step(a_t) ▷ *Execute action*
  $v_H \leftarrow$ AssociateToVertex($\mathcal{G}, o_H$)
  v_H.count $\leftarrow v_H$.count $+ 1$
  $\mathcal{D}_w \leftarrow ((o_t, o_H, a_t, H - t)$ for $(o_t, a_t, t) \in$ trajectory$)$
  return $\mathcal{D}_w, o_H$

function AssociateToVertex($\mathcal{G} = (\mathcal{V}, \mathcal{E}), o_t$)
  $\mathbf{d} \leftarrow$ sort$((\bar{d}_t^v, v)$ for $v \in \mathcal{V})$ ▷ *Predict distances*
  $v, d \leftarrow \mathbf{d}[0]$ ▷ *Associate o_t with nearest vertex*
  return v

function LeastExploredNeighbor($\mathcal{G} = (\mathcal{V}, \mathcal{E}), o_t, \delta_2$)
  $v \leftarrow$ AssociateToVertex($\mathcal{G}, o_t$)
  $\mathcal{V}_n \leftarrow \{v' : \mathcal{E}(v, v') < \delta_2, v' \in \mathcal{V}\}$ ▷ *Retrieve neighbors*
  $\mathbf{c} \leftarrow$ sort$((v'$.count$, v'$.o$)$ for $v' \in \mathcal{V}_n)$
  $v_c, o_c \leftarrow \mathbf{c}[0]$ ▷ *Retrieve neighbor with smallest count*
  return o_c

procedure ExpandGraph($\mathcal{G} = (\mathcal{V}, \mathcal{E}), o_t$)
  $v_t \leftarrow$ Node(count $= 1, o = o_t$) ▷ *Create node for o_t*
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v_t, v_g) : \bar{d}_t^g, g \in \mathcal{V}\}$ ▷ *Add edges*
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{v_t\}$ ▷ *Add vertex*

Layer	Input [Dimensions]	Output [Dimensions]	Layer Details
		Encoder $p_\phi(z \mid o_t, o_g) = \mathcal{N}(\cdot; \mu_p, \Sigma_p)$	
1	o_t, o_g [3, 160, 120]	I_t^g [6, 160, 120]	Concatenate along channel dimensio.
2	I_t^g [6, 160, 120]	E_t^g [1024]	MobileNet Encoder [61]
3	E_t^g [1024]	μ_p [64], σ_p [64]	Fully-Connected Layer, exp activation of σ_p
4	σ_p [64]	Σ_p [64, 64]	torch.diag(σ_p)
		Decoder $q_\theta(a, d \mid o_t, z_t^g) = \mathcal{N}(\cdot; \mu_q, \Sigma_q)$	
1	o_t [3, 160, 120]	E_t [1024]	MobileNet Encoder [61]
2	E_t [1024], z_t^g [64]	$F = E_t \oplus z_t^g$ [1088]	Concatenate image and goal representation
3	F [1088]	μ_q [3], σ_q [3]	Fully-Connected Layer, exp activation of σ_q
4	σ_q [3]	Σ_q [3, 3]	torch.diag(σ_q)
5	μ_q [3]	$\bar{a}_t^g$[2], $\bar{d}_t^g$[1]	Split into actions and distances.

Table 4: Architectural Details of RECON: The inputs to the model are RGB images $o_t \in [0, 1]^{3\times160\times120}$ and $o_g \in [0, 1]^{3\times160\times120}$, representing the current and goal image.