Learning Open-World Robot Navigation from Experience

Dhruv Shah

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-155
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-155.html

August 4, 2024

Learning Open-World Robot Navigation from Experience

by

Dhruv Shah

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Engineering — Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sergey Levine, Chair
Professor Pieter Abbeel

Summer 2024

Learning Open-World Robot Navigation from Experience

Learning Open-World Robot Navigation from Experience

Dhruv Shah

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science (Plan II)**.

Approval for the Report and Comprehensive Examination:

Committee

Sergey Levine

Professor Sergey Levine
Research Advisor

8/3/2024

Date

* * * * *

Professor Pieter Abbeel
Second Reader

8/3/2024

Date

Abstract

Learning Open-World Robot Navigation from Experience

by

Dhruv Shah
Master of Science in Engineering — Electrical Engineering and Computer Sciences

University of California, Berkeley
Professor Sergey Levine, Chair

This report presents a novel approach to long-range robot navigation that combines machine learning with high-level planning. We posit that robust navigation in challenging, real-world environments requires both the ability to learn skills from past experience of the robot, as well as an explicit memory for planning and search. For the former, we describe an algorithm for *experiential learning* for visual navigation, where the robot can learn navigation behaviors directly from its past experience in the real world. For the latter, we design algorithms and systems that combine the low-level learned policy with a high-level topological memory, enabling long-range navigation and exploration in a scalable manner. By combining a learned policy with a topological graph, our system can determine how to reach a visually indicated goal even in the presence of variable appearance and lighting, making it robust for real-world deployment. To reach goals in previously unseen environments, we use a learned latent goal model to learn a density function of reachable future goals and plan over this distribution using the topological memory. Finally, we extend this system so that it can utilize side information, such as schematic roadmaps or satellite imagery, as a planning heuristic. Combining a learned low-level policy with a high-level planner and a learned heuristic allows our learning-based system to navigate kilometer-scale environments in a variety of locations and lighting conditions without any human intervention or collisions. The robotic systems developed in this report serve as a prototype for deploying machine learning models in challenging real-world environments, using a combination of learning and planning.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Robot navigation is one of the most heavily studied topics in robotics [100]. It is often approached in terms of *mapping* and *planning*: constructing a geometric representation of the world from observations, then planning through this model using motion planning algorithms [111, 12, 9]. However, such geometric approaches abstract away significant physical and semantic aspects of the navigation problem that in practice leave a range of real-world situations difficult to handle. These challenges require special handling, resulting in complex systems with many components. Some works have sought to incorporate machine learning techniques to either learn navigational skills from simulation or to learn perception systems for navigation for human-provided labels. In this article, we instead argue that learned navigational models, trained directly on real-world experience rather than human-provided labels or simulators, provide the most promising long-term direction for a general solution to navigation. We refer to such learning approaches as *experiential learning*, because they learn directly from past experience of performing real-world navigation [66]. We will also discuss how this experiential learning paradigm can be combined with high-level planning to build robotic systems that can be robustly deployed in challenging environments.

Geometry-based methods for navigation, based on mapping and planning, are appealing in large part because they simplify the navigation problem into a concise geometric abstraction: if the 3D shape of the environment can be inferred from observations, this can be used to construct an accurate geometric model, a path to the destination can be planned within this model, and that path can then be executed in the real world. However, although some idealized environments fit neatly into this geometric abstraction, real-world settings have a tendency to confound it. Obstacles are not always rigid impassable barriers (e.g., tall grass), and areas that appear geometrically passable might not be (e.g., mud, foliage, etc.). Real-world environments also exhibit patterns that are not used by purely geometric approaches: roads often (but not always) intersect at right angles, city blocks tend to be of equal size, and buildings are often rectangular. Such patterns can lead to convenient shortcuts and intuitive behaviors that are often exploited by humans.

Machine learning can offer an appealing toolkit for addressing these complex situations and exploiting such patterns. In this article, we will focus on the paradigm of *experiential* learning, where a robot learns how to navigate directly from its experience of driving around in the real-world.

Algorithms that learn robotic policies from experience often employ "end-to-end" learning methods [67, 120]. This can either mean that the robot learns the task directly from final task outcome feedback, or that it learns directly from raw sensory perception. Both have appealing benefits, but particularly the former is a critical strength of experiential learning: only by associating actual real-world trajectories with actual real-world outcomes can a robot acquire navigational skills that are not vulnerable to the "leaky abstractions" that afflict other manually designed techniques. For example, the abstraction of geometry doesn't capture that tall grass is traversable. The abstraction of a simulator that doesn't model wheel slip doesn't capture that wheels can become stuck in mud. By learning about real outcomes from real data, such issues can be eliminated.

As we will discuss in subsequent chapters, learned navigation systems can (and should) still employ modularity and compositionality to solve temporally extended tasks. We will argue that effective learning systems, like conventional mapping and planning methods, should still be divided into two parts: a *memory* or "mental map" of their environment, and a high-level *planning* algorithm that uses this mental map to choose a route. Conventional methods simply choose specific abstractions, such as meshes or points in Cartesian space, to represent this map, whereas learning-based methods *learn* a suitable abstraction from data. These learned abstractions are grounded in the things that are actually important for real-world traversability, and they improve as the robot gathers more and more experience in the environment.

The central principle behind experiential learning is to learn from actual experience of attempting (and succeeding or failing) to perform a given task, as opposed to learning from human-provided labels, such as semantic labels provided by humans (e.g., road vs. not road), or demonstrations. Perhaps the best known framework for experiential learning is reinforcement learning (RL) [106], which formulates the problem in terms of learning to maximize reward signals through active online exploration. However, we will make a distinction between the principle of experiential learning – learning how to perform a task using experience – and the *methodology* prescribed by RL. This is because the primary benefits really come from the use of experience, rather than the specific choice of algorithm (RL or otherwise). The particular methods discussed in this report use simple supervised learning methods, though they can be seen as a particularly naïve version of offline RL [68] and could likely utilize more advanced and modern offline RL methods as well.

The goal of this report is to provide a high-level tutorial on how long-range open-world navigational systems can be trained, provide pointers to relevant recent works, and present the overall architecture that a navigational system learned from experience should have. The remainder of this report will focus on building a long-range open-world learned robotic system that can navigate previously unseen kilometer-scale environments, such as suburban neighborhoods and university campuses. We will start by describing the principle of experiential learning, as applied to training goal-reaching navigation policies in Chapter 2. This system will use a combination of end-to-end learning and high-

level planning using a topological memory, and be deployable in challenging real-world environments. We will then extend the capabilities of this system in Chapter 3 by training a data-driven exploration prior to explore novel environments. Lastly, in Chapter 4, we will demonstrate how this robotic system can learn from external cues such as satellite imagery or roadmaps to intelligently *guide* the robot in unseen environments, so that it can explore semantic patterns such as driving on roads and hiking trails.

The robotic system developed in this report serves as a prototype of deploying machine learning models, which can learn from their experience and improve over time, in challenging real-world environments, using a combination of learning and planning. The subsequent chapters of this report contain findings from the following research articles, written over the period of 2020–21:

1. Shah et al., "ViNG: Learning Open-World Navigation with Visual Goals" in *International Conference on Robotics and Automation (ICRA), 2021*
2. Shah et al., "Rapid Exploration for Open-World Navigation with Latent Goal Models" in *Conference on Robot Learning (CoRL), 2021*
3. Shah and Levine, "ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints" in *Robotics: Science and Systems (RSS), 2022*.

2

LEARNING OPEN-WORLD NAVIGATION WITH VISUAL GOALS

Synopsis

We propose a learning-based navigation system for reaching visually indicated goals in a previously seen environment, and demonstrate this system on a real mobile robot platform. By combining a learned policy with a topological graph constructed out of previously observed data, our system can determine how to reach this visually indicated goal even in the presence of variable appearance and lighting. Three key insights, waypoint proposal, graph pruning and negative mining, enable our method to learn to navigate in real-world environments using only offline data, a setting where prior methods struggle. We instantiate our method on a real outdoor ground robot and show that our system, which we call ViNG, outperforms previously-proposed methods for goal-conditioned reinforcement learning. We also study how ViNG generalizes to unseen environments and evaluate its ability to adapt to such an environment with growing experience. Finally, we demonstrate ViNG on a number of real-world applications, such as last-mile delivery and warehouse inspection.

2.1 INTRODUCTION

Visual navigation in complex environments poses several challenges: (i) difficulty in faithfully modeling the complex dynamics and nuanced environmental interactions; (ii) reacting to high-dimensional observations; (iii) cost and safety constraints on collecting data, requiring learning from previously collected (i.e., "offline") experience; and (iv) generalizing effectively across different settings and environments. Planning algorithms achieve many of these desiderata, but their efficacy depends on having the right representation of the task; it remains unclear how to apply many planning algorithms to tasks with image-based observations. On the other hand, humans seemingly have little difficulty navigating complex environments from first-person observations, without GPS or maps, if they have seen the environment before. Humans and animals are known to use "mental maps" that rely on landmarks and other cues [80, 42, 35], and rely heavily on learning. Further, in the absence of spatial positional information (e.g., GPS or maps),

Project website: sites.google.com/view/ving-robot

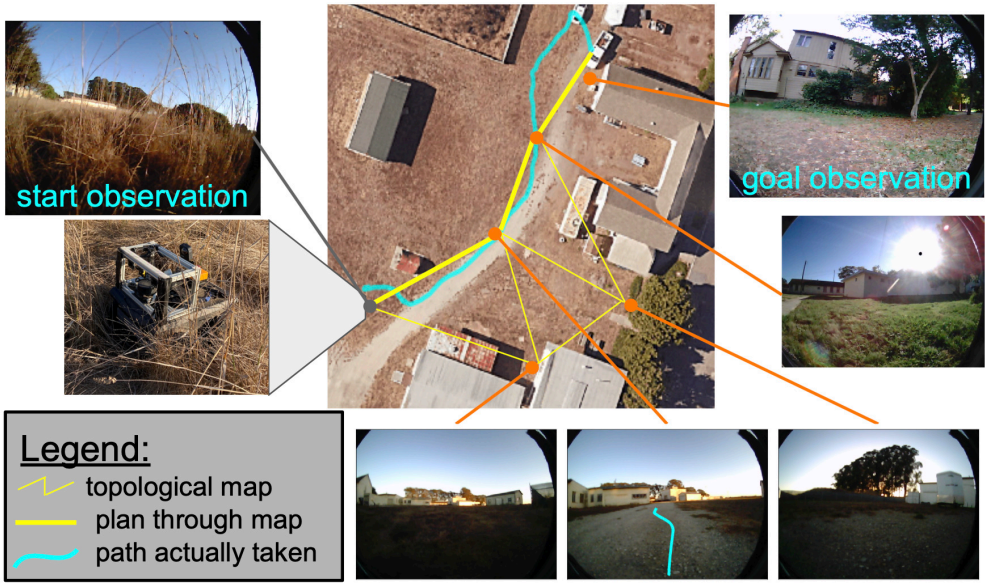


Figure 1: ViNG builds and plans over a *learned* topological graph consisting of previously seen egocentric images, and uses a learned controller to execute the path to a visually indicated goal. Unlike prior work, our method uses purely offline experience and does not require a simulator or online data collection. Note that the graph constructed by our algorithm is not geometric and nodes are *not* associated with coordinates in the world, but only with *image observations* – the top-down satellite image is provided only for visualization and is not available to our method.

specification of a navigational goal itself becomes challenging, since locational goals require the robot to be able to compare its location to the target.

In this chapter, we study learning-based methods for navigation that can similarly utilize graph-structured "mental maps" that are non-geometric in nature, and can enable a robot to navigate in the real-world. We use a natural and intuitive mechanism for specifying goals – where the user provides the robot with a picture of the desired destination. Inspired by humans navigating toward previously seen landmarks, our goal is to enable the robot to navigate to a visually indicated goal. Crucially, such a goal specification scheme does not presume any prior geometric knowledge of the scene, while still providing enough information for the robot to perform the task. Fig. 1 shows an example of such a task.

Towards satisfying these requirements, we present a fully autonomous, self-supervised mobile robot platform for visual goal-reaching in outdoor, unstructured environments which we call ViNG — **Vi**sual **N**avigation with **G**oals. Our approach combines the strengths of dynamical distance learning and graph search. We first learn a function that predicts the *dynamical* distance between pairs of observations, estimating how many time steps are needed to transition between them. We then use this learned dynamical distance to embed past observations into a topological graph, and plan over this graph.

This process makes no geometric assumptions about the environment: reachability is determined entirely by learning from data. Unlike pure planning-based approaches, our method scales to high-dimensional observations and hard-to-model dynamics, and does not assume access to any ground-truth spatial information. Unlike pure learning-based approaches, our method effectively learns from offline experience and reasons over long horizons. Unlike prior methods that combine planning and learning, ViNG learns from offline, real-world data, and does not require a simulator or online data collection.

The primary contribution of this work is a self-supervised robotic system, ViNG, that can efficiently learn goal-directed navigation behaviors in open-world environments without access to spatial maps from an offline pool of data, including randomly collected trajectories. Three key ideas, waypoint proposal, graph pruning and negative mining, differentiate our method from prior work and are critical to the success of our method in this offline setting. ViNG can learn to navigate to an arbitrary user-specified visual goal in a variety of open-world settings, including urban, grassy, and rocky terrain, learning only from offline data. Our experiments show that ViNG learns goal-conditioned behaviors that can effectively plan over long horizons. We show that ViNG outperforms several competitive offline RL and geometric baselines. Further, we show that the learned behaviors transfer to novel environments using as little as 20 minutes of data from the environment and that ViNG can adapt in such novel environments as it gathers more data, resulting in an autonomous, self-improving system. Lastly, we demonstrate two real-world applications enabled by ViNG in dense, urban neighborhoods – last-mile delivery of food or mail, and autonomous inspection of warehouses.

2.2 Related Work

Prior work has studied vision-based mobile robot navigation in many real-world settings, including indoor and outdoor navigation [89, 110, 7], autonomous driving [114, 116], and navigation in extra-terrestrial and underwater environments [59, 23]. The combination of mapping [113] and path planning [63] has been a cornerstone for a number of effective systems [25, 101, 39] and underlies several state-of-the-art navigation systems [5, 2]. Many prior methods make restrictive assumptions, such as access to LIDAR or other structured sensor information and accurate localization, which can limit their suitability for deployment in unstructured environments [24]. Further, prior work often assumes that geometric traversability is faithfully indicated through observations and not misled by (say) non-obstacles such as tall grass [38]. Learning-based systems lift some of these assumptions and can use learned models to perform perception [18, 117], planning [53, 61, 45], or both [67]. In practice, learning temporally extended long-horizon skills with either reinforcement learning (RL) or imitation learning (IL) remains difficult [90, 30].

Recent methods address limitations of the above approaches by combining planning and learning [31, 93, 13, 19, 33, 76]. These methods use learning (i.e., approximate dynamic programming) to solve short-horizon tasks and plan (i.e., use exact dynamic

Figure 2: Challenges with Real-World Navigation: *(Top)* Three observations taken from *exactly the same position* at different times of day exhibit large differences. *(Bottom)* While tall grass and inclined rocks are traversable, a hole filled with dry leaves is not. These examples highlight the challenges with geometric reasoning about traversability.

programming) over non-metric topological graphs [75, 74] to reason over longer horizons. This general approach simultaneously avoids the need for (1) high-fidelity map building and (2) learning temporally-extended behaviors from scratch. However, prior instantiations of this recipe make assumptions that limit their applicability to real-world settings: assuming access to an exact simulation replica of the environment [37, 76], assuming simplified action spaces [31, 93, 13], or requiring online data collection [31, 13]. Our experiments in Section 2.5 demonstrate that prior methods fail when they are not allowed to collect new experience in a simulator or the real-world.

Our method, ViNG, builds on these prior approaches by adding two key ideas: graph pruning and negative sampling. These additional ingredients allow ViNG to lift assumptions made by prior methods: it does not assume access to a simulator, and does not require interactive access to an environment; it is trained using offline, real-world data; and it operates directly on high-dimensional images and predicts continuous actions for the robot. To the best of our knowledge, ViNG is the first system demonstrated on a real-world ground robot that can learn from offline data to reach visually indicated navigational goals over long time horizons without simulated training or hand-designed localization and mapping systems.

2.3 Problem Statement and System Overview

We consider the problem of goal-directed visual navigation: a robot is tasked with navigating to a goal location G given an image observation o_G taken at G. In addition to navigating to the goal, the robot also needs to recognize *when* it has reached the goal, signaling that the task has been completed. The robot does not have a spatial map of the

environment, but we assume that it has access to a small number of trajectories that it has collected previously. This data will be used to construct a graph over the environment using a learned distance and reachability function. We make no assumptions on the nature of the trajectories: they may be obtained by human teleoperation, self-exploration, or a result of a random walk. Each trajectory is a dense sequence of observations $o_1, o_2, \ldots, o_n$ recorded by its on-board camera. 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.

2.3.1 *Mobile Robot Platform*

We implement ViNG on a Clearpath Jackal UGV platform – a small, fast, weatherproof outdoor ground robot ideal for navigating in both urban and off-road environments (see Fig. 1 and 2). 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 camera and an RPLIDAR 2D laser scanner. Inside the Jackal is an NVIDIA Jetson TX2 computer. While the robot carries a GPS and laser scanner, we use these sensors solely as a safety mechanism during data collection. Our method solely operates using images taken from the onboard camera.

2.3.2 *Data Collection & Labeling*

ViNG can learn navigational behaviors from previously-collected, off-policy data – a desideratum of real-world robots. To demonstrate this capability, we run our core experiments using data exclusively from prior work [54]; we also collect a limited amount of additional data for our environment generalization experiments using the same self-supervised data collection strategy. The prior data was collected more than 10 months prior to the experiments in this paper (see Fig. 2 *(top)*), and exhibits significant differences in appearance, lighting, time of year, and time of day as compared to the evaluation setting. This underscores the ability of ViNG to utilize offline data from diverse sources.

2.4 VISUAL NAVIGATION WITH GOALS

We approach the problem of visual goal-conditioned navigation by combining non-metric maps and learned, image-based, goal-conditioned policies. We describe our method in two stages: (i) training two learned functions and (ii) deploying the system, which entails using the learned functions together with past experience to execute goal-directed behavior.

During *training*, we use previously collected experience to learn an environment-independent traversability function $\mathcal{T}$, as well as a relative pose predictor, $\mathcal{P}$. During

Algorithm 1 Training ViNG

Input transitions $\{\tau^{(k)} = (o_1^{(k)}, a_1^{(k)}, o_2^{(k)}, a_2^{(k)}, \cdots)\}_{k=1,\cdots}$
$\mathbb{D}_+ \leftarrow \{(o_i^{(k)}, o_j^{(k)}, d = \min(j-i, d_{\max}))\}_{i \leq j, k=1,\cdots}$
$\mathbb{D}_- \leftarrow \{(o_i^{(k)}, o_j^{(\ell)}, d = d_{\max})\}_{i,j,k \neq \ell}$
Initialize $\mathcal{T}(o_i, o_j)$ and $\mathcal{P}(o_i, o_j)$
while not converged **do**
    $\mathcal{B}_+ \sim \mathbb{D}_+, \mathcal{B}_- \sim \mathbb{D}_-$ ▷ Sample batch.
    $\mathcal{T} \leftarrow$ UpdateDistanceFn$(\mathcal{T}; \mathcal{B}_+ \cup \mathcal{B}_-)$
    get relative pose: $\mathbb{D}_+ \leftarrow \{((o_i^{(k)}, o_j^{(k)}, d_{ij}, p_{ij}\}$
    $\mathcal{P} \leftarrow$ UpdateRelativePoseFn$(\mathcal{P}; \mathcal{B}_+ \cup \mathcal{B}_-)$
end while
return traversability function $\mathcal{T}$, relative pose function $\mathcal{P}$

deployment, the robot builds a topological graph of its environment: a directed graph with vertices as observations and edges encoding traversability and proximity. At each time step t, the robot localizes its current and goal observations (o_t, o_G) in the graph and follows the best path to G, as determined by a graph search algorithm that outputs the next waypoint for the controller. To close the loop, we need a goal-conditioned controller that takes the current and goal observations, and outputs an action a. The controller progressively follows the path directed by the planner until it reaches G.

While the general recipe of ViNG is similar to prior work [93, 76, 31], our experiments demonstrate that two key technical insights contribute to significantly improved performance in the real-world setting: *graph pruning* (Sec. 2.4.2) and *negative mining* (Sec. 2.4.1). Our comparisons to prior methods in Section 2.5 and ablation studies in Section 2.5.4 demonstrate these novel improvements enable ViNG to learn goal-conditioned policies entirely from offline data, avoiding the need for simulators and online sampling, while prior methods struggle to attain good performance, particularly for long-horizon goals.

2.4.1 *Learning Dynamical Distances*

We aim to learn a traversability function $\mathcal{T}(o_i, o_j) \in \mathbb{R}^+$ that reflects whether *any* controller can successfully navigate between observations o_i and o_j. More precisely, we will learn to predict the estimated number of time steps required by a controller to navigate from one observation to another. This function must encapsulate knowledge of physics beyond just geometry. For example, tall grass and bushes might appear visually similar, but grass is compliant and traversable whereas bushes are not. We explored two methods for learning this traversability function: (1) supervised learning and (2) temporal difference learning [107, 52]. To learn the distance function via supervised learning, we create a dataset $\mathbb{D}_+$ of observation pairs (o_i, o_j) taken from the same trajectory and regress to

the number of timesteps $d_{ij} = j - i$ elapsed between these observations. The distance predicted by this approach corresponds to the estimated number of time steps required by the behavior policy (that which collected the experience) when navigating between two observations. Thus, this approach is simple but may overestimate the true shortest path distances.

The second approach to learning the distance function is via temporal difference learning [52]. This approach uses the same experience as before. While this approach adds additional complexity, in theory it converges to the shortest path distance. In our experiments, we found little difference between these two approaches (see Table 2), but expect that the temporal difference learning approach would be important when moving to settings where the shortest path distance is much shorter than a random walk distance.

Negative Mining (Key Idea 1)

In our experiments, we found that training the distance function using only observation pairs from the same trajectory performed poorly. We hypothesize that the root cause was distribution shift: when building the topological graph we must evaluate the distance function on observation pairs collected from different trajectories, possible from different times of day. To mitigate this problem, we augment the dataset by adding a new dataset $\mathbb{D}_-$ obtained by sampling observations from different trajectories, labeled as d_{max}. We find this augmentation, hereby referred to as *negative sampling*, to be critical in the successful training and evaluation of $\mathcal{T}$ in our experiments, offering significant improvements over prior methods.

2.4.2 *The Topological Graph*

We build a topological graph $\mathcal{M}$ using the learned distance function together with a collection of previously-observed observations $\{o_t\}$. Each *node* in the graph corresponds to one of these observations. We add weighted *edges* between every node, using weights predicted by the distance function $\mathcal{T}$.

Graph Pruning (Key Idea 2)

As the robot gathers more experience, maintaining a dense graph of traversability across all observation nodes becomes redundant and infeasible, as the graph size grows quadratically. For our experiments, we sparsify trajectories by thresholding the edges that get added to the graph: edges that are easily traversable $\left(\mathcal{T}(o_i, o_j) < \delta_{\text{sparsify}}\right)$ are not added to the graph, since the controller can traverse those edges with high probability.

Algorithm 2 Deploying ViNG

Input current image o_t, goal image o_G, and topological graph $\mathcal{M}$.
Add o_t, o_G to the map $\mathcal{M}$ using distances from $\mathcal{T}$.
$o_{w_1}, o_{w_2}, \cdots \leftarrow$ Dijkstra(start $= o_t$, goal $= o_G, \mathcal{M}$)
Estimate relative pose of first waypoint: $\Delta p \leftarrow \mathcal{P}(o_t, o_{w_1})$
$u_t \leftarrow$ PD-Controller(Δp)
return control u_t

Planning with the Graph

We localize the current observation o_t and goal observation o_G in the graph, adding direct edges (weighed by their traversability) to their corresponding "most-traversable" neighbors. We use the weighted Dijkstra algorithm to compute the shortest path to goal, and the immediate next node in the planned path is then handed over to the controller.

2.4.3 *Designing the Controller*

After the planner predicts a waypoint observation, the controller must output an action that takes the agent towards that waypoint. The main challenge in navigating to this waypoint is that both the current state and waypoint are represented as high-dimensional observations (e.g., images). To address this challenge, we learn a relative position predictor $\mathcal{P}$ that takes as input two observations and predicts the relative pose between these observations. We learn this relative pose predictor via supervised learning: for pairs of observations (o_i, o_j) that occur nearby within the collected trajectories, we estimate the relative pose Δp_{ij} using onboard odometry and use this relative pose as the label for learning.

The complete controller works as follows. Given the current observation and waypoint observation, we use the relative pose predictor to estimate the relative pose of the waypoint relative to the robot's current position. The robot then uses odometry and a simple PD controller to steer toward this waypoint. We compare against alternative controllers in Section 2.5.4.

2.4.4 *Implementation Details*

Inputs to the traversability function $\mathcal{T}$ and relative pose predictor $\mathcal{P}$ are pairs of observations of the environment, represented by a stack of two consecutive RGB images obtained from the onboard camera at a resolution of 160×120 pixels. $\mathcal{T}$ comprises a MobileNet encoder [49] followed by three densely connected layers to project the $1024-$dimensional latents to 50 class labels. $\mathcal{P}$ has a similar architecture as $\mathcal{T}$, comprising of a MobileNet

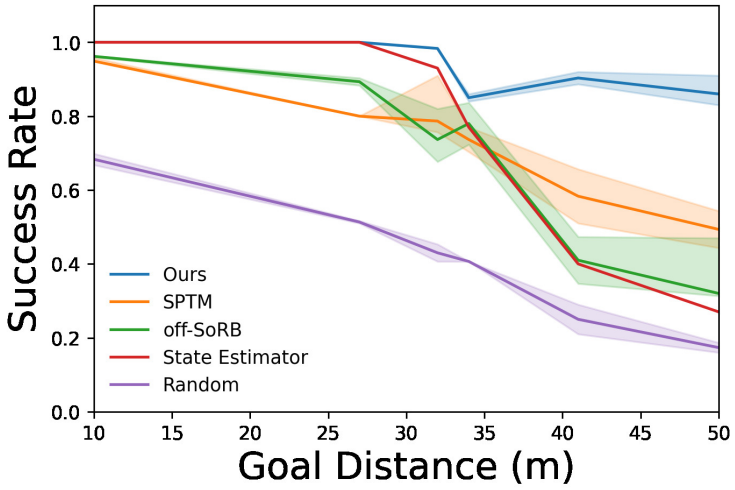


Figure 3: Real-World Navigation: While all non-random methods successfully reach nearby goals, only ViNG reaches goals over 40 meters away. Here, success rate is defined as the average over portion of the expert trajectory to goal that each run successfully completes.

encoder followed by three densely connected layers projecting the 1024−dimensional latents to 3 outputs for waypoints: $\{\Delta x, \Delta y\}$. Both $\mathcal{T}$ and $\mathcal{P}$ use the same encoder.

We train the traversability function on $\mathbb{D}_+ \cup \mathbb{D}_-$, discretizing the timesteps d_{ij} into bins $\{1, \cdots, d_{\text{max}} = 50\}$ and minimizing the cross entropy loss. The relative pose predictor $\mathcal{P}$ is trained on $\mathbb{D}_+$ to minimize the ℓ_2 regression loss. We use a batch size of 128 and perform gradient updates using the Adam optimizer [56] with learning rate $\lambda = 10^{-4}$. Algorithms 1 and 2 summarize our approach in the training and deployment stages, respectively.

2.5 EXPERIMENTS

We designed our experiments to answer three questions:

Q1. How does ViNG compare to prior methods for the task of goal-conditioned visual navigation from offline data?

Q2. Does ViNG generalize to novel environments? Can it adapt on the fly?

Q3. What are the alternate design choices for the controller and how do they compare against our choice in Section 2.4.3?

2.5.1 *Goal-Conditioned Visual Navigation from Offline Data*

We perform our evaluation in a real-world outdoor environment consisting of urban and off-road terrain. We train on 40 hours of data that was gathered in prior work [54] over 10 months prior to the experiments in this paper. The data shows significant variation in appearance due to seasonal changes (see Fig. 2); learning navigational affordances and traversability would require the algorithms to discard the irrelevant modes of variance (e.g., appearance) and establish correspondence across seasons and times of day.

Since this evaluation takes place in the real world, we do not have the luxury of training online RL policies or transfer from simulation. We evaluate ViNG against four baselines:

SPTM: a dense topological graph combined with a controller that maps observation pairs to motor commands, trained via supervised learning [93]

off-SoRB: an offline variant of SoRB that uses a topological graph and offline RL to learn a distributional Q-function [31]

State Estimator: a naïve baseline that uses a state estimator network that regresses observations to ground-truth state (x, y, θ), followed by a position controller; note that this baseline has access to true position (from GPS), which is not available to our method

Random: a random walk, as described in Section 2.4.3

While there have been other successful instantiations of methods combining planning and learning, they make some limiting assumptions that make them difficult to apply to our problem setting. *LSTN* [76] uses a photorealistic simulator to train its distance and action models, using $\sim$ 1.5M samples, while *PRM-RL* [37] uses a 3D kinematic simulator simulation replica to train a reactive controller, coupled with physical rollouts in the real world to build a PRM. ViNG does not assume access to any simulator, and learns directly from offline real data.

Towards answering Q1, we evaluate the goal-reaching performance of ViNG. We select 6 {start, goal} image pairs in the original urban environment and compare the goal reaching performance of each method (avg. of 3 trials). We report the success metric as the average over portion of the expert trajectory to goal that each run successfully completes.

As shown in Fig. 3, ViNG performs well on all tasks, achieving a success rate of 86% on even the most challenging tasks. As expected, the random baseline, which ignores the goal, fails to reach most goals. The state estimator baseline performs a bit better, but struggles to reach more distant goals because it is not reactive, and hence cannot take actions to avoid collisions. Off-SoRB performs well on nearby goals, but as the goals get increasingly difficult to reach, it is unable to follow the planned trajectory. Visualizing the

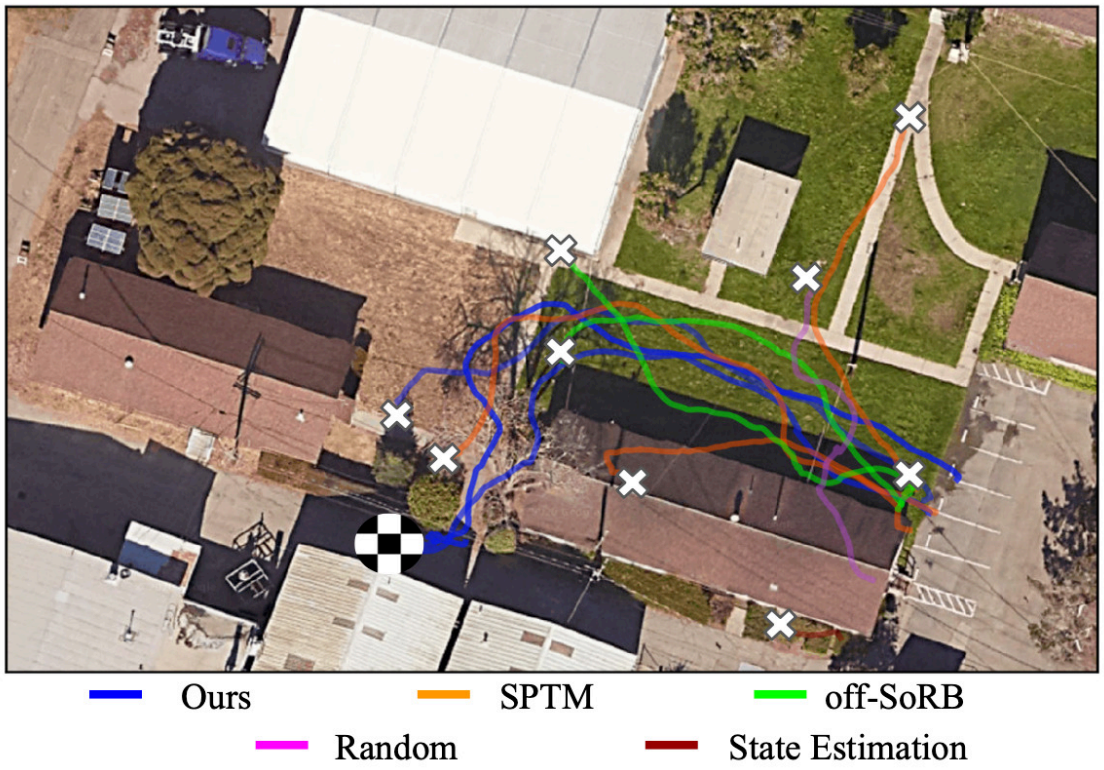


Figure 4: Qualitative Results in the urban **Environment**: Each approach was directed to a visual goal $\sim$ 50m away (marked by checkerboard circle) – with 3 runs per approach. ViNG is the only approach that is consistently able to reach the goal while avoiding collisions or getting stuck.

topological graph built by SoRB uncovers many disconnected components, resulting in no path to goal. We hypothesize that this is attributed to the difficulty in training Q-functions from offline data. SPTM, which uses supervised learning instead of Q-learning, is effective at solving the task on shorter horizons and outperforms off-SoRB on longer horizons. However, ViNG still performs substantially better on all goal distances, especially those over 30 meters. We attribute these improvements to the additional negative sampling and graph pruning techniques discussed in Section 2.4. We visualize trajectories in Fig. 4.

2.5.2 *Generalization and Adaptation*

The experiments in the previous section evaluate navigation to new goals in a previously seen environment. In this section, we additionally evaluate how quickly ViNG can adapt to an entirely new, unseen environment, by constructing a new graph and finetuning the models. We use the four settings shown in Fig. 5, all of which are distinct from the setting used in our main experiments (Sec. 2.5.1). In each new environment, a human controlled the robot to provide initial exploration data. After this initial data collection,

Environment	ViNG Source	ViNG Target	ViNG Finetune	SPTM Finetune
barracks	0.27	0.42	**0.96**	0.74
industrial	0.13	0.44	**0.84**	0.68
park	0.04	0.32	**0.82**	0.71
tall grass	0	0.38	**0.79**	0.56

Table 1: Generalization Results: Our approach to generalization ("ViNG -Finetune") successfully navigates learns to navigate in four new environments (shown in Fig. 5) using just 60 minutes of experience in the new environment. Baselines that use only experience from the source or target domains are substantially less successful. Applying our finetuning approach on top of SPTM shows some generalization, but is outperformed by ViNG-Finetune.

the robot collected experience *autonomously*: it randomly sampled a previously-observed image as thegoal and used ViNG to attempt to reach this goal. After each episode, we used all experience from the new environment (both the expert trajectories and the self-collected trajectories) to finetune $\mathcal{T}$ and $\mathcal{P}$. We refer to this approach to generalization as ViNG -Finetune.

In Fig. 5 we visualize trajectories after 60 min of data collection in the new environment and observe that the robot successfully reaches the goal in most cases. We emphasize that these environments are considerably different from those used in Sec. 2.5.1, on which our models were initially trained. To illustrate the learning dynamics in this generalization setting, we plot self-collected rollouts after 0 minutes, 20 minutes, and 60 minutes of practice in the new environments. As shown in Fig. 6, the robot's performance in the new domain gets progressively better with more (autonomous) practice; after 60 minutes it succeeds in reaching the goal in all three attempts.

Table 1 summarizes the success rate on the generalization task of our method and two alternative versions of ViNG. ViNG-Source directly uses the traversability function and relative pose function trained in the source domain (Sec. 2.5.1), without incorporating any experience from the new environment. In contrast ViNG-Target learns these same models using only experience from the new "target" domain, without leveraging any of the previously-collected experience. ViNG-Finetune outperforms these baselines, highlighting the importance of combining old and new experience. As an additional baseline, we take the SPTM model from Sec. 2.5.1 and finetune it on experience from the new domain. We observe that ViNG -Finetune also generalizes better than SPTM-Finetune, We hypothesize that ViNG generalizes better than SPTM because of the additional hierarchical structure of ViNG.

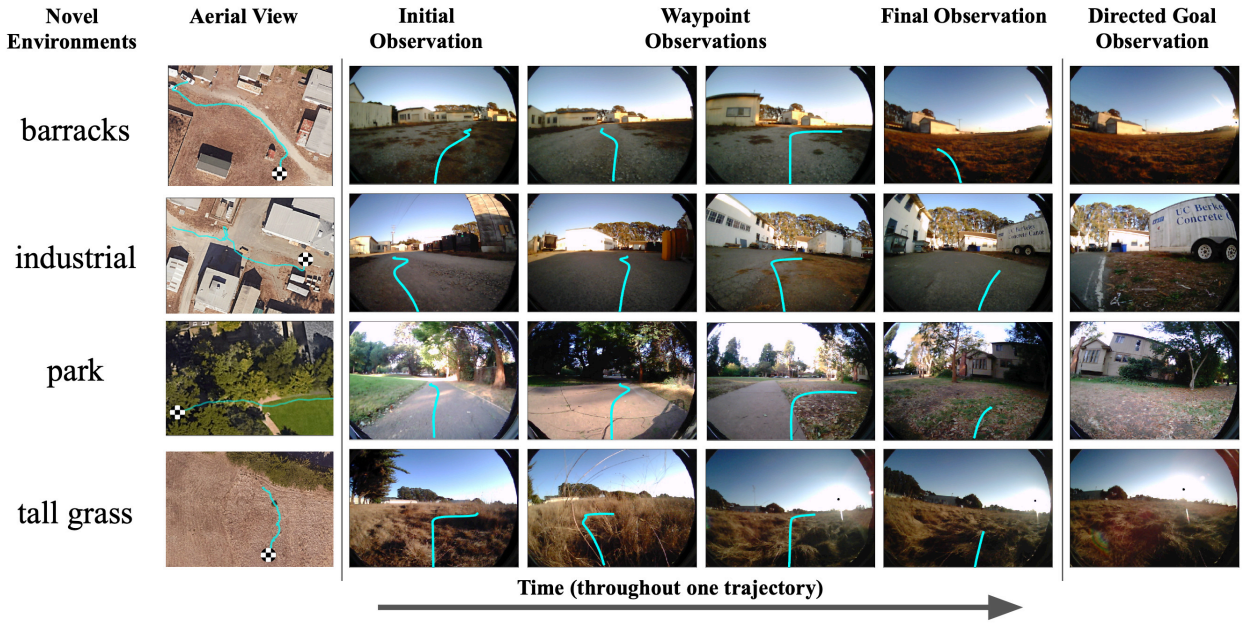


Figure 5: Generalization Experiments: We evaluate ViNG in four new outdoor environments. For each, we collect a few dozen minutes of experience to adapt the distance function and relative pose predictor. Then, given a goal image (last column, checkerboard location in aerial view), the robot attempts to navigate to the goal. Columns $4-7$ indicate that the robot succeeds in reaching the goal image. Cyan lines indicate the actions taken by ViNG.

2.5.3 *Comparisons to Online Methods*

While Section 2.5.1 establishes that ViNG outperforms competitive offline methods for the task of goal-conditioned navigation, here we also investigate the performance of our method in comparison to popular online RL algorithms. Since the sample complexity of online RL algorithms forbids us from testing this in the real world, we use a Unity-based photorealistic outdoor navigation simulator. We include new additional baselines in the simulated experiment:

PPO: a popular reactive controller for indoor visual navigation algorithms [96, 119]

SoRB: online version of the "off-SoRB" baseline [31]

We show results in Fig. 7. PPO performs poorly and is outperformed by ViNG, suggesting that a single image-based reactive policy is insufficient for solving long-horizon goal-reaching tasks, even when given access to 200 hours of online experience. SoRB outperforms other baselines and performs on par with ViNG. However, whereas ViNG requires 40 hours of offline data, SoRB requires 200 hours of online data, and must recollect this data for every experiment.

Figure 6: Fast Adaptation to a New Environment: After training ViNG in one environment, we deploy the system in a novel environment, shown above. By practicing to reach self-proposed goals and using that experience to finetune the controller, ViNG is able to quickly gain competence at reaching distance goals in this new environment, using just 60 minutes of experience. Example rollouts towards a goal 35m away (marked by checkerboard circle) demonstrate ViNG self-improving from interactions in the barracks environment.

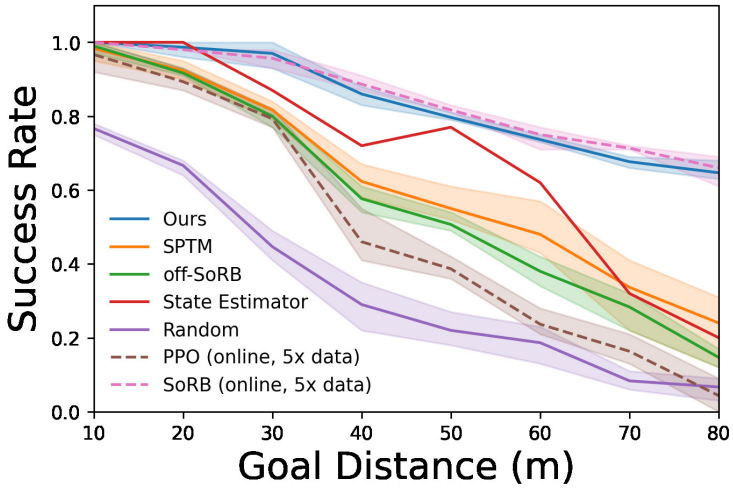


Figure 7: Results from Simulated Navigation: ViNG is substantially more successful at reaching distance goals than all offline baselines, while performing competitively with SoRB, a popular online baseline combining Q-learning and topological graphs. We emphasize that SoRB and PPO require 5× online data collection, making them prohibitively expensive to apply in the real-world.

2.5.4 *Ablation Experiments*

A key design decision for ViNG that differentiates it from prior methods (e.g., [76, 31]) is how the controller generates actions to reach the next waypoint. We evaluate variants of ViNG that use alternative controllers and present results in Table 2. Two simple baselines, "direct actions" and "direct actions (discrete)", use the goal-conditioned behavior cloning method of [93, 27] to directly predict (discrete) actions from the current and goal observations, without utilizing the topological map. Recall that our method uses the planner to command waypoints and then uses the relative pose together with a PD controller to reach each waypoint. We compared against a baseline that uses a different low-level controllers to reach these same waypoints: "Waypoint, Discrete" takes actions using the "direction actions (discrete)" controller described above. As an alternative training scheme, "TD Waypoint" is a variant of our method that learns the traversability function via TD learning instead of supervised learning. Finally, we compare to two ablations of our method that skip the graph pruning and negative sampling stages of ViNG.

2.5.5 *Applications and Qualitative Results*

ViNG's ability to navigate using perception and landmarks, without access to maps or localization, can enable a number of intuitive applications, which we illustrate through

Controller	Success Rate @ Distance d (m)				
	$d=10$	$d=20$	$d=30$	$d=40$	$d=50$
Direct Actions (Discrete)	0.87	0.81	0.74	0.65	0.45
Direct Actions	0.98	0.89	0.74	0.73	0.4
Waypoint, Discrete	**1.0**	0.95	0.91	0.82	0.7
Waypoint	**1.0**	**1.0**	**0.95**	0.88	0.81
TD Waypoint	**1.0**	**1.0**	**0.96**	**0.87**	**0.87**
Waypoint, No Pruning	**1.0**	**0.88**	0.81	0.79	0.52
Waypoint, Only Positives	**1.0**	0.91	0.75	0.76	0.43

Table 2: Ablation Experiments: We investigate design choices for the parametrization of the controller. Using waypoints as a mid-level action space is key to the performance of ViNG, which is particularly emphasized for distant goals. While training the models, we show that ViNG can be trained with either supervised or TD learning and report similar performance. We also show that the two key ideas presented – graph pruning and negative sampling – are indeed essential for the performance of ViNG in the real-world.

qualitative results in this section. We constructed two demonstrations that reflect potential applications of our system:

1. *Contactless Last-Mile Delivery*: We demonstrate last-mile delivery in a residential complex by using ViNG to autonomously deliver mail and food to visually-indicated delivery locations. In this setting, users specify delivery destinations for the robot simply by taking a photograph of the desired destination, and the robot autonomously navigates to this destination to deliver a package.

2. *Autonomous Inspection*: Densely constructed building complexes, like university campuses, are often unmapped or lack accurate spatial localization. We reprogram ViNG to periodically navigate to landmarks, specified as images, around the campus to set up an autonomous patrolling system. Discrepancies can be identified by comparing the observations to previous observations (stored in the topological graph).

Figures 8 and 9 show ViNG successfully performing these tasks in the urban environment. Videos of the qualitative results, generalization experiments, and real-world applications can be found at the project website (sites.google.com/view/ving-robot).

2.6 DISCUSSION

In this paper, we proposed ViNG: a system for goal-directed navigation using visual observations and goals on an outdoor ground robot. While conceptually similar to

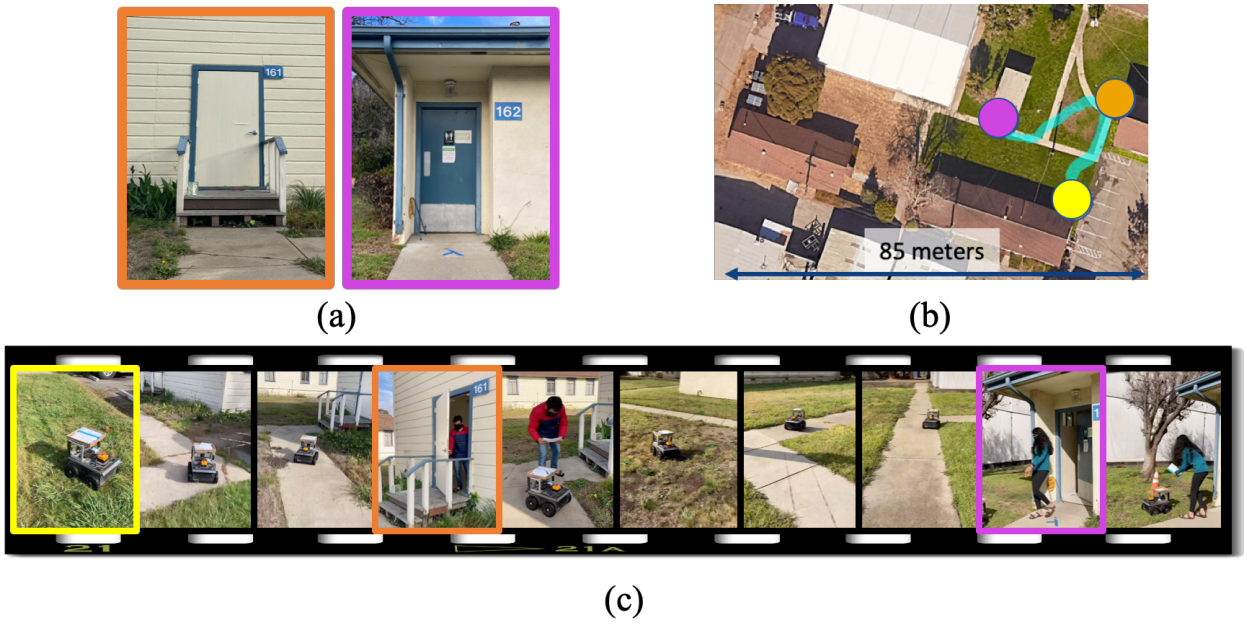


Figure 8: Contactless Last-Mile Delivery Demo: Given a set of visually-indicated goals (a), ViNG can perform contactless delivery in the urban neighborhood successfully, as shown in the filmstrip (c). An overhead view (b) with starting position marked in yellow and respective goals marked in orange and magenta shows the trajectory of the robot (cyan). *Note: The satellite view (b) is solely for visualization and is not available to the robot.*

prior methods, we demonstrate that a few key design choices, such as pruning the topological graph, parametrizing the controller in terms of a relative pose predictor and sampling negatives while training to minimize distribution shift, allow ViNG to learn to successfully navigate using only offline experience, a setting in which many prior methods fail. Intriguingly, we also demonstrate that ViNG can be quickly adapted to navigate in new environments. These generalization and self-improvement attributes highlight that learning-based approaches are not only an effective mechanism for handling high-dimensional observations, but are also amenable to fast adaptation to novel environments. Further, we have demonstrated ViNG on a number of real-world applications in dense, urban environments that may be unmapped or GPS-denied, and specifying visual goals is convenient – contactless last-mile delivery and autonomous inspection.

Our method requires a static, offline dataset of observations over which we can plan. Many real-world tasks are non-stationary, with the distribution of observations shifting over time (e.g., lighting changes, dynamic objects, etc.). In future work, we aim to incorporate representations of observations and goals that are robust to such distributional shifts, which would expand the generalization capabilities of our method.

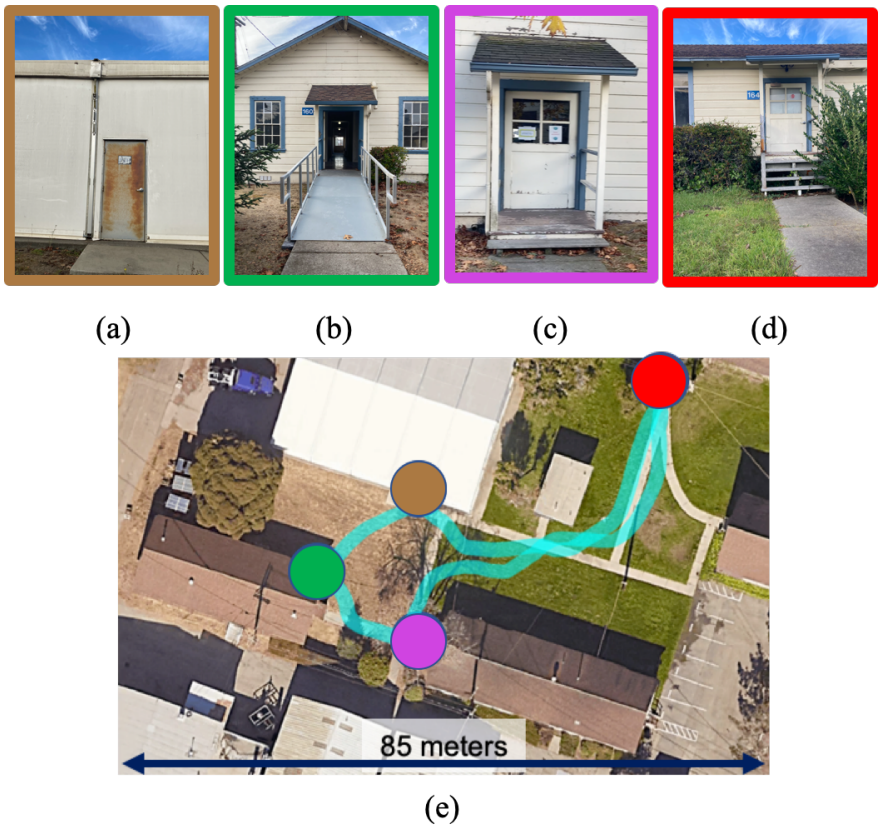


Figure 9: Autonomous Inspection Demo: Given a set of visual landmarks (a–d) in a university campus, ViNG can perform autonomous inspection by navigating to these goals periodically. An overhead view (b) shows color-coded goals and the trajectory taken by robot (cyan) in one cycle. *Note: The satellite view (e) is solely for visualization and is not available to the robot.*

ACKNOWLEDGMENTS

This research was funded by the Office of Naval Research, DARPA Assured Autonomy, and ARL DCIST CRA W911NF-17-2-0181, with computing support from Google and Amazon Web Services. The authors would like to thank Jonathan Fink and Ethan Stump for their help setting up the simulation environment used for developing this research.

3

OPEN-WORLD EXPLORATION WITH LATENT GOAL MODELS

Synopsis

In this chapter, we build upon our robotic learning system by enabling it to perform autonomous exploration and navigation in diverse, open-world environments. Our core idea is to use 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.

3.1 INTRODUCTION

Robustness is a key challenge in learning to navigate diverse, real-world environments. A robotic learning system must be robust to the difference between an offline training dataset and the real world (i.e., it must generalize), be robust to non-stationary changes in the real world (i.e., it must ignore visual distractors), and be equipped with mechanisms to actively explore to gather information about traversability. Different environments may 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

Project website: sites.google.com/view/recon-robot

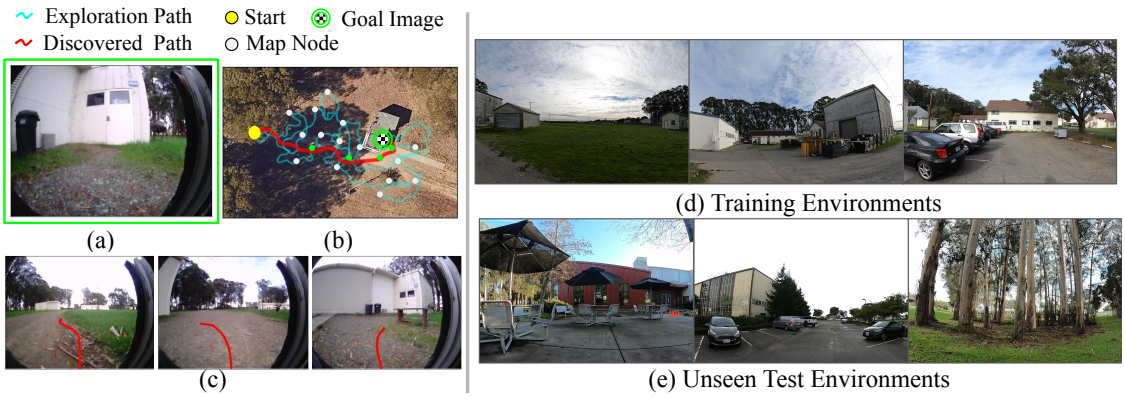


Figure 10: System overview: Given a goal image *(a)*, RECON explores the environment *(b)* by sampling prospective *latent* goals and constructing a topological map of images (white dots), operating only on visual observations. After finding the goal *(c)*, RECON can reuse the map to reach arbitrary 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)*.

affordances. Our approach to this problem uses 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.

We demonstrate our method, **R**apid **E**xploration **C**ontrollers for **O**utcome-driven **N**avigation (RECON), on a mobile ground robot and evaluate against 6 competitive baselines spanning over 100 hours of real-world experiments in 8 distinct open-world environments (Fig. 10). 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. We make this dataset publicly available as a source of real-world interaction data for future resesarch.

3.2 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 [60, 8, 57, 108], while others use use global strategies based on the frontier method [121, 15, 47]. 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 [93, 34, 31, 99, 76], 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 [41, 71]. 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 [104, 84, 4, 11, 92, 94, 13], information gain [48, 77], or methods based on counts, densities, or distance from previously-visited states [6, 46, 65]. However, these methods learn to reason about the novelty of a state only after visiting it. Recent works [102, 14] 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 [72, 58], 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 [29, 91, 78] and transfer learning [109, 64, 83, 44, 40]. Our method uses an information bottleneck [115], which serves a dual purpose: first, it provides a representation that can aid the generalization capabilities of RL algorithms [50, 43], and second, it serves as a measure of task-relevant uncertainty [3], 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 [52, 95, 86, 32] and imitation learning (IL) [28, 41, 71, 105, 103, 88]. 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 [87, 21, 123, 85, 69]. Unlike these prior methods, however, our method makes use of previous experience in *different* environments to accelerate learning in the current environment.

3.3 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.

3.3.1 *Mobile Robot Platform*

We implement RECON on a Clearpath Jackal UGV platform. 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 [54], 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.

3.3.2 *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. 10 (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[1] and provide further details in Appendix A.1.

[1] Available for download at sites.google.com/view/recon-robot/dataset.

3.4 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.

3.4.1 *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 [3, 1]. We use a context-conditioned representation of goals to learn a control policy in the target environment. 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 [3], we approximate the intractable objective in Eq. 1 with a variational posterior and decoder (an upper bound), resulting in the maximization objective:

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

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.

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.

Algorithm 3 *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: $\quad\quad o_n \leftarrow \text{LeastExploredNeighbor}(\mathcal{G}, o_t; \delta_2)$
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: $\quad\quad$ **end if**
13: $\quad\quad \mathcal{D}_w, o_t \leftarrow \text{SubgoalNavigate}(z_t^w; H)$
14: $\quad\quad \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_w$
15: $\quad\quad \text{ExpandGraph}(\mathcal{G}, o_t)$
16: $\quad\quad$ Step $L(\phi, \theta; \mathcal{D}, \beta)$ for γ epochs ▷ Eq. 2
17: $\quad$ **end while**
18: $\quad$ **return** networks p_ϕ, q_θ and graph $\mathcal{G}$
19: **end function**

3.4.2 *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, where we have a latent goal model pre-trained on the offline dataset. Given a subgoal, our algorithm (Alg. 3) proceeds by executing actions towards the subgoal for a fixed number of timesteps (Alg. 3 L13). The data collected during subgoal navigation expands the topological memory (Alg. 3 L15) and is used to fine-tune the model (Alg. 3 L16). 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 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. 3 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.

(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. 3 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. 3 L11).

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

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 A.2.1. We also share broader implementation details including choice of hyperparameters, model architectures and training details in Appendix A.2.2.

Method	Expl. Time (mm:ss) ↓	Nav. Time (mm:ss) ↓	SCT [122] ↑
PPO + RND [11]	21:18	00:47	0.22
InfoBot [43]	23:36	00:48	0.21
Active Neural SLAM (ANS) [13]	21:00	00:45	0.33
ViNG [99]	19:48	00:34	0.60
Ours + Episodic Curiosity (ECR) [94]	14:54	00:31	0.73
RECON (Ours)	**09:54**	**00:26**	**0.92**

Table 3: 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 [102] 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.

3.4.3 *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. 3.4.1) 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.1).

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. 3 and Sec. 3.4.2.

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. 4.

3.5 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?

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?

3.5.1 *Goal-Directed Exploration in Novel Environments*

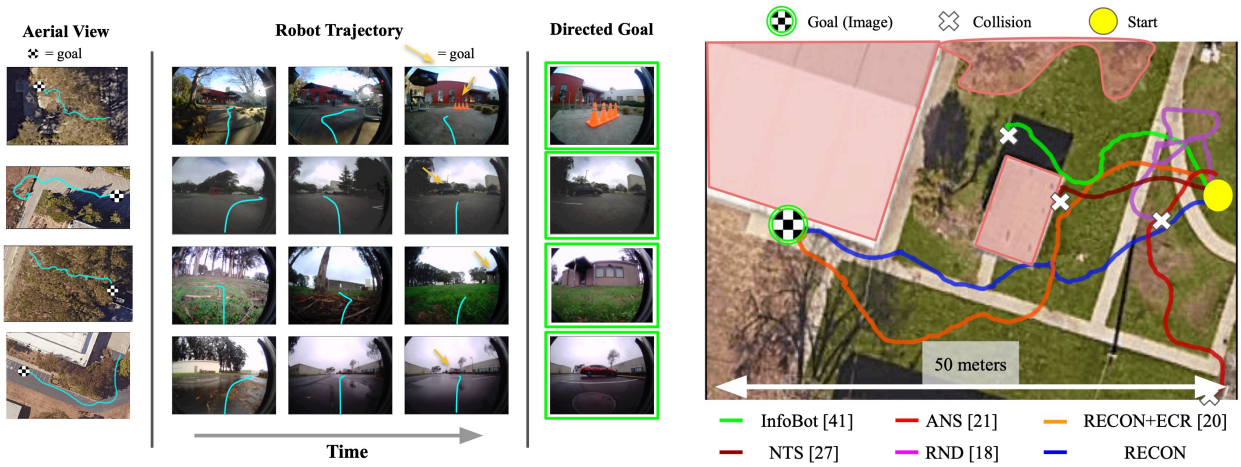


Figure 11: Visualizing goal-reaching behavior of the system: *(left)* Example trajectories to goals discovered by RECON in *previously unseen* environments. *(right)* Policies learned by the different methods in one such environment. Only RECON and ECR reach the goal successfully, and RECON takes the shorter route.

We perform our evaluation in a diverse variety of outdoor environments (examples in Fig. 10), including parking lots, suburban housing, sidewalks, and cafeterias. We train our self-supervised navigation model using an offline navigation dataset (Sec.3.3.2) 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 five 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 [11], which we use with PPO [96, 119]. 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 [43], 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 [13]. We adapt it to the goal-directed task by using the distance function from RECON to detect when the goal is nearby.

4. **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 [99].

5. **Episodic Curiosity (ECR):** A method that executes random action sequences at the frontier of a topological graph for exploration [94]. We implement this as an *ablation* of our method that samples random action sequences, rather than rollouts to sampled goals (Alg. 3 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 [122]. We show quantitative results in Table 3, and visualize sample trajectories of RECON and the baselines in Fig. 11.

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 [43, 13, 102]. We attribute RECON's success to the context-conditioned sampling strategy (described in Sec. 3.4.1), which proposes goals that can accelerate the exploration of new environments.

We then study RECON's ability to quickly reach goals after initial discovery. Table 3 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. 11 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.

3.5.2 *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 RECON to 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. 12-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. 12 shows trajectories taken by the robot as it successfully navigates to the goal in scenarios with varying obstacles and lighting conditions. These results 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).

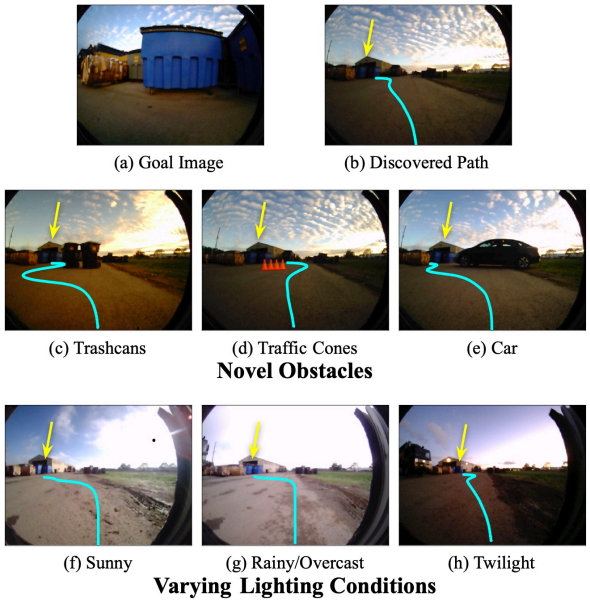


Figure 12: Exploring non-stationary environments: The learned representation and topological graph is robust to visual distractors, enabling reliable navigation to the goal under novel obstacles *(c–e)* and appearance changes *(f–h)*.

3.5.3 *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

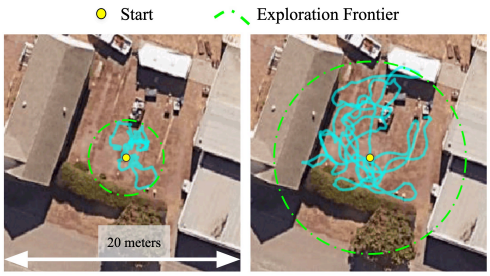


Figure 13: Exploration via sampling from our context-conditioned prior (*right*) allows the robot to explore 5 times faster than using random actions, e.g. in ECR [94] (*left*).

Method	Expl. Time ↓	Nav. Time ↓	SCT [122] ↑
Reactive	11:54	00:37.4	0.63
Rand. Actions	14:54	00:31.4	0.73
V. Sampling	14:06	00:28.7	0.83
Ours	**09:56**	**00:25.8**	**0.92**

Table 4: Ablation experiments confirm the importance of using an information bottleneck and a non-parametric memory.

the environment. We compare the coverage of trajectories of the robot over 5 minutes of exploration when: *(a)* it executes random action sequences [94], 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. 13).

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. 3.5.1.
- *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 4. These results corroborate the observations in Fig. 13: 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.

3.6 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.

ACKNOWLEDGMENTS

This research was supported by ARL DCIST CRA W911NF-17-2-0181, DARPA Assured Autonomy, and the Office of Naval Research. The authors would like to thank Suraj Nair and Brian Ichter for useful discussions, and Gregory Kahn for setting up the infrastructure used for autonomous collection of real-world data.

4

KILOMETER-SCALE EXPLORATION WITH GEOGRAPHIC HINTS

Synopsis

In this chapter, we extend our robotic learning system so it can utilize side information such as schematic roadmaps, satellite maps and GPS coordinates as a planning heuristic, to achieve kilometer-scale robot navigation and exploration in previously unseen environments. Our method, ViKiNG, incorporates a local traversability model, which looks at the robot's current camera observation and a potential subgoal to infer how easily that subgoal can be reached, as well as a heuristic model, which looks at overhead maps for hints and attempts to evaluate the appropriateness of these subgoals in order to reach the goal. These models are used by a heuristic planner to identify the best waypoint in order to reach the final destination. Our method performs no explicit geometric reconstruction, utilizing only a topological representation of the environment. Despite having never seen trajectories longer than 80 meters in its training dataset, ViKiNG can leverage its image-based learned controller and goal-directed heuristic to navigate to goals up to 3 kilometers away in previously unseen environments, and exhibit complex behaviors such as probing potential paths and backtracking when they are found to be non-viable. ViKiNG is also robust to unreliable maps and GPS, since the low-level controller ultimately makes decisions based on egocentric image observations, using maps only as planning heuristics.

4.1 INTRODUCTION

Robotic navigation has conventionally been approached as a geometric problem, where the robot constructs a 3D model of the environment and then plans a path through this model. End-to-end learning-based methods offer an alternative approach, where the robot learns to correlate observations with traversability information directly from experience, without full geometric reconstruction [124, 18, 53]. This can be advantageous

Project website: sites.google.com/view/viking-release

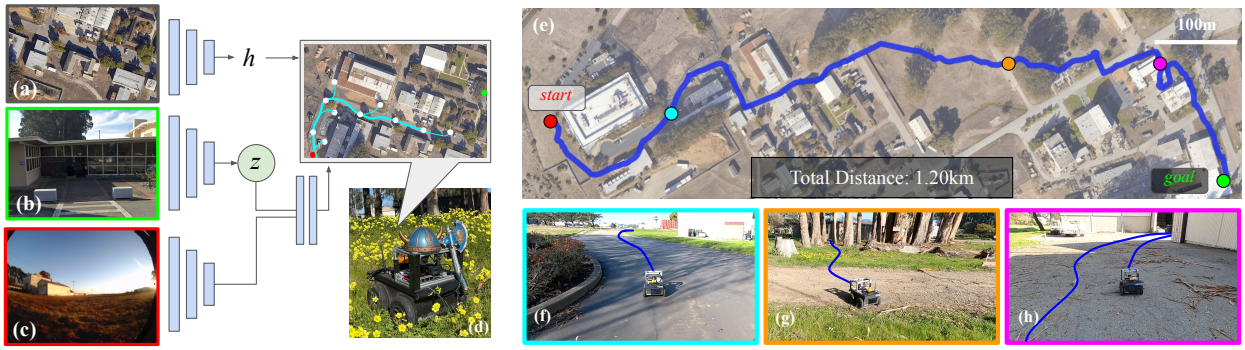


Figure 14: Kilometer-scale autonomous navigation with ViKiNG: Our learning-based navigation system takes as input the current egocentric image (c), a photograph of the desired destination (b), and an overhead map (which may be a schematic or satellite image) (a) that provides a *hint* about the surrounding layout. The robot (d) uses learned models trained in *other* environments to infer a path to the goal (e), combining local traversability estimates with global heuristics derived from the map. This enables ViKiNG to navigate *previously unseen* environments (e), where a single traversal might involve following roads (f), off-road driving under a canopy (g), and backtracking from dead ends (h).

because, in many cases, geometry alone is neither necessary nor sufficient to traverse an environment, and a learning-based method can acquire patterns that are more directly indicative of traversability, for example by learning that tall grass is traversable [54] while seemingly traversable muddy soil should be avoided. More generally, such methods can learn about common patterns in their environment, such as that houses tend to be rectangular, or that fences tend to be straight. These patterns can lead to common-sense inferences about which path should be taken through an unknown environment even before that environment has been fully mapped out [79].

However, dispensing with geometry entirely may also be undesirable: the spatial organization of the world provides regularities that become important for a robot that needs to traverse large distances to reach its goal. In fact, when humans navigate new environments, they make use of both geographic knowledge, obtained from overhead maps or other cues, and learned patterns [118]. But in contrast to SLAM, humans don't require maps or auxiliary signals to be very accurate: a person can navigate a neighborhood using a schematic that roughly indicates streets and houses, and reach a house marked on it. Humans do not try to accurately reconstruct geometric maps, but use approximate "mental maps" that relate landmarks to each other topologically [36]. Our goal is to devise learning-enabled methods that similarly make use of geographic *hints*, which could take the form of GPS, roadmaps, or satellite imagery, without requiring these signals to be perfect.

We consider the problem of navigation from raw images in a *novel* environment, where the robot is tasked with reaching a user-designated goal, specified as an egocentric

image, as shown in Figure 14. Note that the robot has *no prior experience* in the target environment.The robot has access to geographic side information in the form of a schematic roadmap or satellite imagery, which may be outdated, noisy, and unreliable, and approximate GPS. This information, while not sufficient for navigation by itself, contains useful cues that can be used by the robot. The robot also has access to a large and diverse dataset of experience from *other* environments, which it can use to learn general navigational affordances. We posit that an effective way to build such a robotic system is to combine the strengths of machine learning with informed search, by incorporating the geographic hints into a learned heuristic for search. The robot uses approximate GPS coordinates and an overhead map as geographic side information to help solve the navigation task, but does not assume that this information is particularly accurate—resembling a person using a paper map, the robot uses the GPS localization and an overhead map as *hints* to aid in visual navigation. Note that while we do assume access to GPS, the measurements are only accurate up to 2-5 meters (4-10$\times$ the scale of the robot), and cannot be used for local control.

The primary contribution of this work is ViKiNG, an algorithm that combines elements of end-to-end learning-based control at the low level with a higher-level heuristic planning method that uses this image-based controller in combination with the geographic hints. The local image-based controller is trained on large amounts of prior data from *other* environments, and reasons about navigational affordances directly from images without any explicit geometric reconstruction. The planner selects candidate waypoints in order to reach a faraway goal, incorporating the geographic side information as a planning heuristic. Thus, when the hints are accurate, they help the robot navigate toward the goal, and when they are inaccurate, the robot can still rely on its image observations to search through the environment. We demonstrate ViKiNG on a mobile ground robot and evaluate its performance in a variety of open-world environments not seen in the training data, including suburban areas, nature parks, and a university campus. Our local controller is trained on 42 hours of navigational data, and we test our complete system in 10 different environments. Despite never seeing trajectories longer than 80 meters in its training data, ViKiNG can effectively use geographic side information in the form of overhead maps to reach user-specified goals in *previously unseen environments* over 2 kilometers away in under 25 minutes.

4.2 RELATED WORK

Robotic navigation has been studied from a number of different perspectives in different fields. Classically, it is often approached as a problem of geometric mapping or reconstruction followed by planning [112]. In unknown environments, the mapping problem can be formulated in terms of information gain with local strategies [8, 57, 108], global strategies based on the frontier method [121, 47, 15, 16], or by sampling "next-best views" [22, 82, 97], but such methods typically aim to map or reconstruct an entire

environment, rather than achieve a single navigational goal. Active exploration methods have sought to modify this by jointly incentivizing an exploration objective along with reconstruction of the map [26, 70]. Both the goal-directed and mapping-focused methods aim to reconstruct the geometry of their environment, and do not directly benefit from training with prior data. Some approaches have sought to incorporate learning into mapping and reconstruction [73, 51], which benefits from prior data, but still aims at dense geometric reconstruction. Our approach uses a model that is trained with data from prior environments to predict *traversability* rather than geometry, and this model is then used in combination with geographic hints to plan a path to the goal.

In this respect, ViKiNG is also related to prior work on learning-based navigation, which is often formulated in terms of the "PointGoal" task [72]. Many such works rely on simulation and reinforcement learning, utilizing millions (or billions) of online trials to train a policy [61, 119]. In contrast, our method learns entirely from previously collected offline data, extrapolates to significantly longer paths than it is trained on, and does not require any simulation or online RL.

A number of prior learning-enabled methods also combine learned models with graph-based planning, using a topological graph to represent the environment [93, 10, 34, 99, 76]. These methods often assume access to data from the test environment to start with a viable graph, which may not be available in a new environment. Some works have studied this unseen setting by predicting explorable areas for semantically rich parts of the environment to accelerate visual exploration [102, 14]. While these methods can yield promising results in a variety of domains, they come at the cost of high sample complexity (over 10M samples) [72], making them difficult to use in the real world—the most performant algorithms take 10-20 minutes to find goals up to 50m away [98].

The closest prior work to ViKiNG is by Shah et al. (RECON) [98], which uses a learned representation over feasible subgoals to uniformly explore the environment. Like RECON, our method trains a local model that predicts temporal distances and actions for nearby subgoals, and then incorporates this model into a search procedure that incrementally constructs a topological graph in a novel environment. However, in contrast to RECON, which performs an uninformed search, ViKiNG incorporates geographic hints in the form of approximate GPS coordinates and overhead maps. This enables ViKiNG to reach faraway goals, up to 25× further away than the furthest goal reported by RECON, and to reach goals up to 15× faster than RECON when exploring a novel environment.

4.3 Visual Navigation with Geographic Hints

Our aim is to design a robotic system that learns to use first-person visual observations to reach user-specified landmarks, while also utilizing geographic *hints* in the form of approximate GPS coordinates and overhead maps. At the core of our approach is a deep neural network that takes in the robot's current camera observation o_t, as well as an observation o_w of a potential subgoal w (we use "subgoal" and "waypoint"

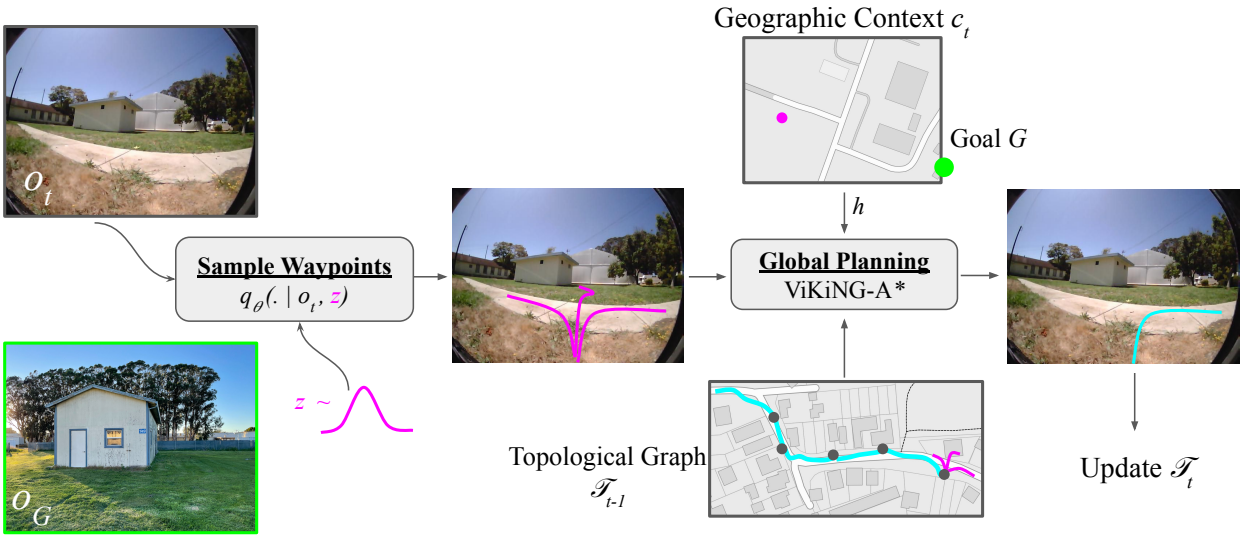


Figure 15: An overview of our method. ViKiNG uses latent subgoals z proposed by a learned low-level controller, which operates on raw image observations o_t, for global planning on a topological graph $\mathcal{T}$ to reach a distant goal o_G, indicates by a photograph and an approximate GPS location. A learned heuristic parses the overhead image c_t to bias this search towards the goal.

interchangeably), and predicts the time to reach w (or "temporal distance"), the best current action to do so, and the resulting spatial offset in terms of GPS coordinates. This model can also sample latent representations of potential *reachable* waypoints from the current observation o_t, which are used as candidate subgoals for planning. The model is trained on large amounts of data from a variety of training environments and, when the robot is placed in a *new* environment that it has not seen, it is used to incrementally construct a *topological* (non-geometric) graph to navigate to a distant user-specified goal. This goal is indicated by a photograph with an approximate GPS coordinate, and may be several kilometers away. The learned model alone is insufficient to navigate to such a distant goal in one shot, and therefore our planner uses a combination of the model's predictions and geographic information to plan a sequence of subgoals that search for a path through the environment, incrementally constructing the graph.

This process corresponds to a kind of heuristic search, where the geographic side information provides a heuristic to bias the robot to explore towards the goal as it constructs the topological graph. The latent goal model is used to determine reachability in this topological graph, and the geographic heuristic is used to steer the graph by exploring the environment. In a novel environment, the robot must incrementally build this graph using *physical search*, by visiting new nodes and expanding its frontier. The decision about *where* to actually go is determined by the first-person images, and the

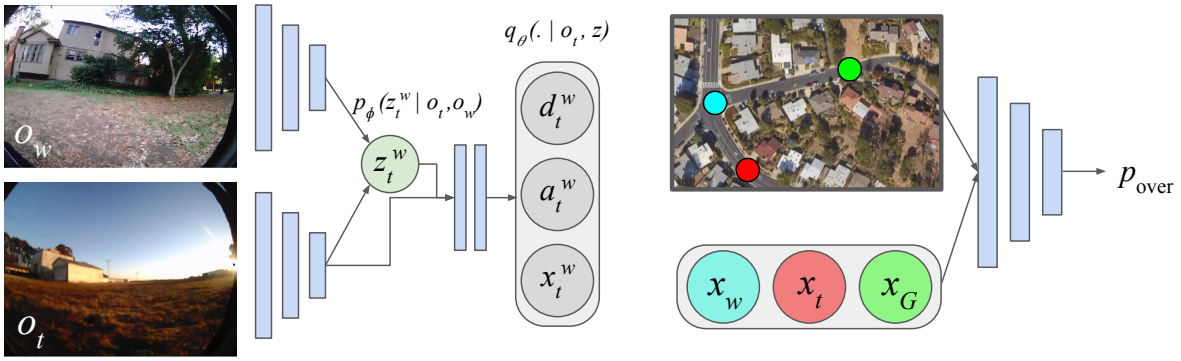


Figure 16: The learned models used by ViKiNG. The latent goal model (left) takes in the current image o_t. It also takes in either a true waypoint image o_w, or samples a *latent* waypoint $z_t^w \sim r(z_t^w)$ from a prior distribution, and then predicts, its temporal distance from o_t (d_t^w), the action to reach it (a_t^w), and its approximate GPS offset (x_t^w). The heuristic model (right) takes in an overhead image c_t, the approximate GPS coordinates of the current location (x_t) and destination (x_G), and the coordinates of the waypoint inferred by the latent goal model (x_w), and predicts an approximate heuristic value of the waypoint w for reaching the final destination.

geographic information is used only as a heuristic, allowing ViKiNG to remain robust to noisy or unreliable side information. We overview our method in Figure 15.

4.3.1 *Low-level Control with a Latent Goal Model*

Our low-level model maps the current image observation o_t and a waypoint observation o_w to: (1) the temporal distance d_t^w to reach w from o_t; (2) the first action a_t^w that the robot must take now to reach w; (3) a prediction of the (approximate) offset in GPS readings between o_t and w, x_t^w. (1) and (3) will be used by the higher-level planner, and (2) will be used to drive to w, if needed. We would also like this model to be able to *propose*, in a learned latent space, potential subgoals w that are reachable from o_t, and predict their corresponding values of d_t^w, a_t^w, and x_t^w.

We present the model in Figure 16, with precise architecture details in the supplementary materials. The model is trained by sampling pairs of time steps in the trajectories in the training set. For each pair, the earlier time step image becomes o_t, and the later image becomes o_w. The number of time steps between them provides the supervision for d_t^w, the action taken at the earlier time step supervises a_t^w, and the later GPS reading is transformed into the coordinate frame of the earlier time step to provide supervision for x_t^w. The model is trained via maximum likelihood. Note that by training the model on data in this way, we not only enable it to evaluate reachability of prospective waypoints, but also make it possible to inherit behaviors observed in the data. For example, in our experiments, we will show that the model has a tendency to follow sidewalks and

forest trails, a behavior it inherits from the portion of the dataset that is collected via teleoperation.

Besides predicting d_t^w, a_t^w, and x_t^w, our planner requires this model to be able to *sample* potential reachabale waypoints from o_t (see Figure 15). We implement this via a variational information bottleneck (VIB) inside of the model that bottlenecks information from o_w. Thus, the model can either take as input a real image o_w of a prospective waypoint, or it can sample a *latent* waypoint $z_t^w \sim r(z_t^w)$ from a prior distribution. We train the model so that sampled latent waypoints correspond to feasible locations that the robot can reach from o_t without collision.

Training the latent goal model: The full model, illustrated in Figure 16, can be split into three parts: a waypoint encoder $p_\phi(z_t^w|o_w, o_t)$, a waypoint prior $r(z_t^w)$, and a predictor $q_\theta(\{a, d, x\}_t^w|z_t^w, o_t)$. The latent waypoint representation z_t^w can either be sampled from the prior (which is fixed to $r(z_t^w) \triangleq \mathcal{N}(0, I)$), or from the encoder $p_\phi(z_t^w|o_w, o_t)$ if a waypoint image o_w is provided. This latent waypoint is used together with o_t to predict all desired quantities according to $q_\theta(\{a, d, x\}_t^w|z_t^w, o_t)$. The training set consists of tuples $(o_t, o_w, \{a, d, x\}_t^w)$, but the model must be trained so that samples $z_t^w \sim r(z_t^w)$ also produce valid predictions. We accomplish this by means of the VIB [3], which regularizes the encoder $p_\phi(z_t^w|o_w, o_t)$ to produce distributions that are close to the prior $r(z_t^w)$ in terms of KL-divergence. We refer the reader to prior work for a derivation of the VIB [3], and present our training objective for p_ϕ and q_θ below:

$$\begin{aligned}\mathcal{L}_{\text{VIB}}(\theta, \phi) = E_{\mathcal{D}}[&- \mathbb{E}_{p_\phi}\left[\log q_\theta\left(\{a, d, x\}_t^w \mid z_t^w, o_t\right)\right] \\ &+ \beta \text{KL}\left(p_\phi(z_t^w \mid o_w, o_t)||r(z_t^w)\right)] \end{aligned} \tag{3}$$

The outer expectation over all tuples $(o_t, o_w, \{a, d, x\}_t^w) \in \mathcal{D}$ in the training distribution is estimating using the training set. The first term causes the model to accurately predict the desired information, while the second term forces the encoder to remain consistent with the prior, which makes the model suitable for sampling latent waypoints according to $z_t^w \sim r(z_t^w)$. As the encoder p_ϕ and decoder q_θ are conditioned on o_t, the representation z_t^w only encodes *relative* information about the subgoal from the context—this allows the model to represent *feasible* subgoals in new environments, and provides a compact representation that abstracts away irrelevant information, such as time of day or visual appearance. An analogous representation has been proposed in prior work [98], but did not predict spatial offsets and was used only for *uninformed* exploration without geographic hints.

4.3.2 *Informed Search on a Topological Graph*

The model described above can effectively reach nearby subgoals, for example those on which the robot has line of sight, but we wish to reach goals that are more than a kilometer away. To reach distant goals, we combine the model with a search procedure that incorporates geographic hints from satellite images or roadmaps. The system does

Algorithm 5 *ViKiNG-A* for Physical Search*

function ViKiNG-A*(start S, goal info o_G, x_G)
  $\Omega \leftarrow \{S\}$
  while Ω not empty **do**
    $w_t \leftarrow \min(\Omega, f)$
    DriveTo(w_t) ▷ update visitations v on the way
    observe image o_t
    add w_t to graph $\mathcal{T}$ ▷ use $q_{\theta,\phi}$ on o_t to get distances
    if close(o_t, o_G) **finish** ▷ use $q_{\theta,\phi}(\{a, d, x\}_t^w | o_t, o_G)$
    remove w_t from Ω
    sample waypoints w near w_t (Section 4.3.1)
    for each w sampled near w_t **do**
      if not contains(Ω, w) **then** add w
    end for
    for each waypoint $w \in \Omega$ **do**
      $f(w) = g(t, w) + d^w_{\Pr[w]} + h(w) + v(\Pr[w])$
    end for
  end while
  return failure
end function

not require this information to be accurate, instead using it as a planning heuristic while still relying on egocentric camera images for control. Our high-level planner plans over a topological graph $\mathcal{T}$ that it constructs incrementally using the low-level model in Section 4.3.1 as a local planner. We first describe a generic version of the algorithm for any heuristic, and then describe the data-driven heuristic function that we extract from the geographic hints via contrastive learning.

Challenges with *physical search*: Our "search" process involves the robot physically searching through the environment, and is not purely a computational process. In contrast to standard search algorithms (e.g., Dijkstra, A*, IDA*, D*, etc.), each "step" of our search involves the robot driving to a subgoal and updating the graph. Standard graph search algorithms assume (i) the ability to *visit* any arbitrary node, and (ii) access to a set of *neighbors* for every node and the corresponding "edge weight," before visiting each neighbor. Physical search with a robot violates these assumptions, since robots cannot "teleport" and *visiting* a node incurs a driving cost. Furthermore, the real world does not provide "edge weights" and the robot needs to estimate the cost to reach an unvisited node *before* actually driving to it.

An algorithm for informed physical search: To solve these challenges, we design ViKiNG-A*, an A*-like search algorithm that uses our latent goal model and a learned heuristic to perform *physical search* in real-world environments. While ViKiNG-A* does prefer

shorter paths, it does not aim to be optimal (in contrast to A*), only to reach the goal successfully. We will use a heuristic $h(w)$, fully described in the next section, which we assume provides a *comparative* evaluation of candidate waypoints in terms of their anticipated temporal distance to the destination. Algorithm 5 outlines ViKiNG-A*.

Like A*, ViKiNG-A* maintains a priority queue "open set" Ω of unexplored fringe nodes and a "current" node that represents the least-cost node in this set, which we refer to as w_t. It also maintains a graph with visited waypoints, $\mathcal{T}$, where nodes correspond to images seen at those nodes, and edges correspond to temporal distances estimated by the model in Section 4.3.1. At every iteration, the robot *drives* to the least-cost node in the open set (L5), using a procedure that we outline later. When it reaches w_t, it observes the image o_t using its camera (L6). This allows it to add o_t to the graph $\mathcal{T}$ (L7), connecting it to other nodes by evaluating the distances using the model in Section 4.3.1. The graph construction is analogous to prior work [99, 98]. If the robot is close to the final goal image o_G according to the model (L8), the search ends. o_t also allows it to sample nearby candidate waypoints using the model in Section 4.3.1 (L10): first sampling $z_t^w \sim r(z_t^w)$ from the prior, and then decoding distances d_t^w, a_t^w, and x_t^w, from which it can compute absolute locations as $x_w = x_t + x_t^w$. Each sampled waypoint is stored in the open set, and annotated with the current image o_t and d_t^w. We refer to w_t as the *parent* of w, and index it as $\Pr[w]$. Note that we do *not* have access to the image o_w, as we have not visited the sampled waypoint w yet, and therefore we must store the current image o_t instead. This also means that we *cannot* connect these waypoints to the graph $\mathcal{T}$ except through their parent. Next, we re-estimate the cost of each waypoint in the open set, including the newly added waypoints.

The cost for each waypoint $w \in \Omega$ from the current point w_t consists of four terms (L15): (1) $g(t, w)$, the cost to navigate to the *parent* of w, which is part of the graph $\mathcal{T}$; this can be computed as a shortest path on the graph $\mathcal{T}$, and is zero for the current node. (2) $d^w_{\Pr[w]}$, the distance from the parent of w to w itself. (3) $h(w)$, the heuristic cost estimate of reaching the final goal from w (see Section 4.3.3). (4) $v(\Pr[w])$, the visitation count of $\Pr[w]$, computed as $CN(\Pr[w])$, where C is a constant and $N(\Pr[w])$ is a count of how many times the robot drove to $\Pr[w]$ via the DriveTo subroutine; this acts as a *novelty bonus* to encourage the robot to explore novel states, a strategy widely used in RL [62, 6]. Summing these terms expresses a preferences for nodes that are fast to reach from w_t (1 + 2), get us closer to the goal (3), and have not been heavily explored before (4). At the next iteration (L4), the robot picks the lowest-cost waypoint and again drives to it.

To navigate to a selected waypoint w (DriveTo), the robot employs a procedure analogous to prior work on learning-based navigation with topological graphs [99, 98], planning the shortest path through $\mathcal{T}$, and selecting the next waypoint on this path. Once the waypoint w is selected, the model $q_{\theta,\phi}(\{a, d, x\}_t^w | o_t, o_w)$ is used to repeatedly choose the action a_t^w based on the current image o_t, until the distance d_t^w becomes small, indicating that the waypoint is reached and the robot can navigate to the next waypoint (in practice, it's convenient to replan the path at this point, as is standard in MPC). Each

time the DriveTo subroutine reaches a node, it also increments its count $N(w)$ which is used for the novelty bonus $v(w)$. The helper function close uses the model in Section 4.3.1 to check if the estimated temporal distance d_t^w is less than ϵ for two observations, and the contains operation on a set checks if a given node is *close* to any node inside the set. These modifications allow A*-like operations on the nodes of our graph, which are continuous variables.

4.3.3 *Learning a Goal-Directed Heuristic for Search*

We now describe how we extract a heuristic h from geographic side information. As a warmup, first consider the case where we only have the GPS coordinates for a waypoint (x_w) and final goal (x_G). We can use $\|x_g - x_w\|$ as a heuristic to bias the search to waypoints in the direction of the goal, and this heuristic can be readily obtained from the model in Section 4.3.1. However, we would like to compute the heuristic function using some side information c_t, such as a roadmap or satellite image, that does not lie in a metric space. Thus, we need to *learn* the heuristic function from data. Since ViKiNG-A* does not aim to be optimal (only seeking a feasible path), we do not require the heuristic to be admissible.

We train the heuristic $h_{\text{over}}(x_w, x_G, x_t, c_t)$ to score the favorability of a sampled candidate waypoint w for reaching the goal G from current location x_t, given side information c_t. In our case, c_t is an overhead image that is roughly centered at the current location of the robot. Our heuristic is based on an estimator for the probability $p_{\text{over}}(w \to G | x_w, x_G, x_t, c_t)$ that a given waypoint w lies on a valid path to the goal G. We use the same training set as in Section 4.3.1 to learn a predictor for p_{over}. Given p_{over}, we can generate a heuristic $h_{\text{over}} := \lambda_{\text{over}}(1 - p_{\text{over}})$ to steer ViKiNG-A* towards the goal (Alg. 5 L14). Note that, since we evaluate the heuristic for sampled candidate waypoints, we do not have access to x_w, but we can predict it by using the model in Section 4.3.1 to infer the offset x_t^w using o_t and the sampled latent code, and then calculate x_w from x_t and x_t^w. Thus, the heuristic is technically a function of c_t, o_t, x_t, and x_G.

Our procedure for training $p_{\text{over}}(w \to G | x_w, x_G, x_t, c_t)$ is based on InfoNCE [81], a contrastive learning objective that can be seen as a binary classification problem between a set of *positives* and *negatives*. At each training iteration, we sample a random batch $\mathcal{B}$ of sub-trajectories k from our training set, where x_S is the start of k and x_E is the end, and c_S is an overhead image centered at x_S. We sample a positive example by picking a random time step in this subtrajectory, and using its position x_{w^+}. The negatives x_{w^-} are locations of other randomly sampled time steps from other trajectories, comprising the set $\mathcal{W}^-$. In this way, we train a neural network model to represent $p_{\text{over}}(w \to G | x_w, x_G, x_t, c_t)$ (see Figure 16, right) via the InfoNCE objective:

$$\mathcal{L}_{\text{NCE}} = -\mathbb{E}_{\mathcal{B}} \left[\log \frac{p_{\text{over}}(w^+ \to E | x_{w^+}, x_E, x_S, c_S)}{\sum_{w^- \in \mathcal{W}^-} p_{\text{over}}(w^- \to E | x_{w^-}, x_E, x_S, c_S)} \right] \tag{4}$$

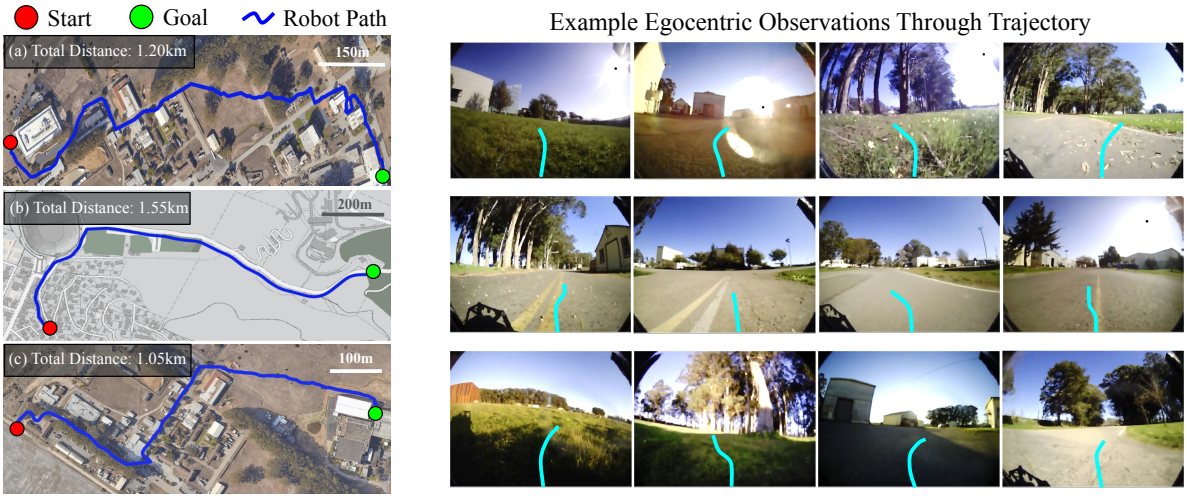


Figure 17: Examples of kilometer-scale goal-seeking in *previously unseen* environments using only egocentric images (right) and a schematic roadmap or satellite image as *hints* (left). ViKiNG can navigate in complex environments composed of roads, meadows, trees and buildings.

This heuristic can only reason about waypoints and goals at the scale of individual trajectories in the training set (up to 50m). For kilometer-scale navigation, the heuristic needs to make predictions for goals that are much further away, so we take inspiration from goal chaining in reinforcement learning [17] and combine overlapping trajectories in the training set (according to GPS positions) into larger trajectory groups. For a batch $\mathcal{B}$ of trajectories, we combine two trajectories if they intersect in 2D space. The resulting macro-trajectories thus have multiple start and goal positions, and can extend for several kilometers. We then sample the sub-trajectories for x_S, x_E, and x_{w^+} from these much longer macro-trajectories, giving us positive examples between very distant x_S, x_E pairs. This allows p_{over} to be trained on a vast pool of long-horizon goals and improves the reliability of the heuristic. We provide more details about this procedure in Appendix B.1.

4.4 VIKING IN THE REAL WORLD

We now describe our experiments deploying ViKiNG in a variety of real-world outdoor environments for kilometer-scale navigation. Our experiments compare ViKiNG to other learning-based methods, evaluate its performance at different ranges, and study how it responds to degraded or erroneous geographic information.

4.4.1 *Mobile Robot Platform*

We implement ViKiNG on a Clearpath Jackal UGV platform (see Fig. 14). 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. Under open skies, the GPS unit is accurate up to 2-5 meters, which is 4-10× the size of the robot. In addition, we added a forward-facing 170° field-of-view RGB camera. Compute is provided by an NVIDIA Jetson TX2 computer, and a cellular hotspot connection provides for monitoring and (if necessary) teleoperation. Our method uses only the monocular RGB images from the onboard camera, unfiltered measurements from onboard GPS, and overhead images (roadmap or satellite) queried at the current GPS location, without any other processing.

4.4.2 *Offline Training Dataset*

Our aim is to leverage data collected in a wide range of different environments to (i) enable the robot to learn navigational affordances that generalize to novel environments, and (ii) learn a global planning heuristic to steer physical search in novel environments. To create a diverse dataset capturing a wide range of navigation behavior, we use 30 hours of publicly available robot navigation data collected using an autonomous, randomized data collection procedure in office park style environments [98]. We augmented this dataset with another 12 hours of teleoperated data collected by driving on city sidewalks, hiking trails, and parks. Notably, ViKiNG never sees trajectories longer than 80 meters, but is able to leverage the learned heuristic (Section 4.3.3) to reach goals over a kilometer away at over 80% of the average speed in the training set. The average trajectory length in the dataset is 45m, whereas our experiments evaluate runs in excess of 1km. The average velocity in the dataset is 1.68 m/s, and the average velocity the robot maintains in testing is 1.36 m/s. We provide more details about the dataset in Appendix B.2.

4.4.3 *Kilometer-Scale Testing*

For evaluation, we deploy ViKiNG in a variety of *previously unseen* open-world environments to demonstrate kilometer-scale navigation. Figure 17 shows the path taken by the robot in search for a user-specified goal image and location. ViKiNG is able to utilize geographic hints, in the form of a roadmap or satellite image centered at its current position, to steer its search of the goal. In a university campus (Fig. 17(a, c)), we observe that the robot can identify large buildings along the way and plan around it, rather than following a greedy strategy. Since the training data often contains examples of the robot driving around buildings, ViKiNG is able to leverage this prior experience and generalize to novel buildings and environments. On city roads (Fig. 17(b)), the learned heuristic shows preference towards following the sidewalks, a characteristic of the training data in city environments. It is important to note that while the robot has seen some prior data on sidewalks and in suburban neighborhoods, it has never seen the specific areas (see Appendix B.2 for further details). For videos of our experiments, please check out our project page.

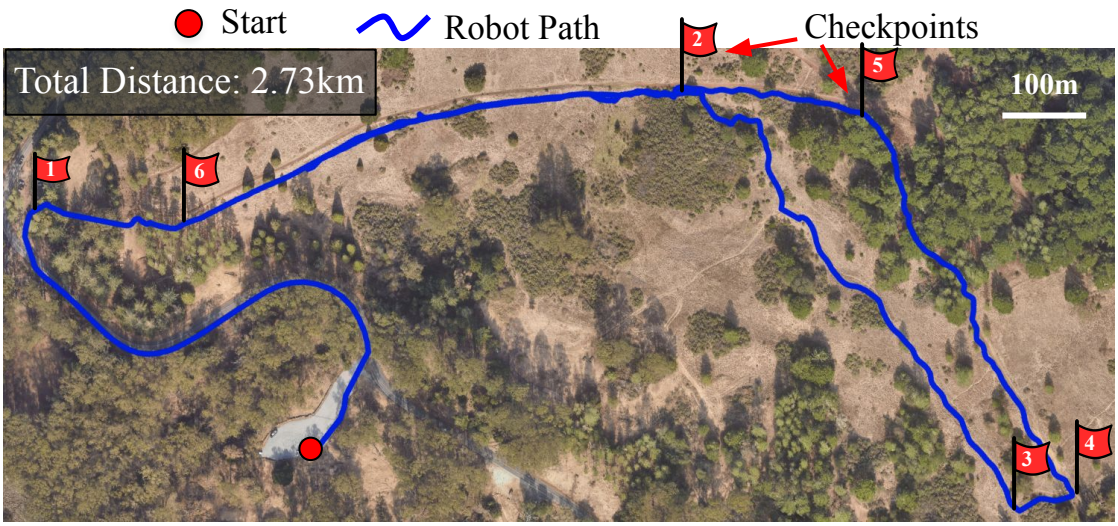


Figure 18: ViKiNG can follow a sequence of goal checkpoints to perform search in complex environments, such as this 2.73km hiking trail.

These long-range experiments also exhibit successful backtracking behavior—when guided into a cul-de-sac by the planner, ViKiNG turns around and resumes its search for the goal from another node in the "openSet", reaching the goal successfully (see Figure 14(h)). While the learned heuristic provides high-level guidance, the local control is done solely from first person images. This is illustrated in Figure 14(g), where the robot navigates through a forest, where the satellite image does not contain any useful information about navigating under a dense canopy. ViKiNG is able to successfully navigate through a patch of trees using the image-based model described in Section 4.3.1. We can also provide ViKiNG with a set of goals to execute in a sequence to provide more guidance about the path (e.g., an inspection task with landmarks), as demonstrated in the next experiment.

A hiking ViKiNG: We deploy ViKiNG, with access to satellite images as hints, on a 2.7km hiking trail with a 70m elevation gain by providing a sequence of six checkpoint images and their corresponding GPS coordinates. Algorithmically, we run ViKiNG-A* on every goal (one at a time) while reusing the topological graph $\mathcal{T}$ across goals. Figure 18 shows a top-down view of the path taken by the robot—ViKiNG is able to successfully combine the strengths of a learned controller for collision-free navigation with a learned heuristic that utilizes the satellite images to encourage on-trail navigation between checkpoints. Since the offline dataset contains examples of trail-following, the robot learns to stay on trails when possible. This behavior is emergent from the data—there is no other mechanism that encourages staying on the trails, and in several cases, a straight-line path between the goal waypoints would not stay on the trail (e.g., the first checkpoint in Figure 18).

Figure 19: ViKiNG can utilize a satellite image to follow a sequence of visual landmarks (top) in complex suburban environments, such as this 2.65km loop stretching across buildings, meadows and roads.

Autonomous visual inspection: We further deploy ViKiNG in a suburban environment for the task of visual inspection specified by five images of interest. ViKiNG is able to successfully navigate to the landmarks by using satellite imagery, traveling a distance of 2.65km without any interventions. Figure 19 shows the specified images and a top-down view of the path taken by the robot on the trail.

4.4.4 *Quantitative Evaluation and Comparisons*

We compare ViKiNG to four prior approaches, each trained using the same offline data as our method. All methods have access to the egocentric images, GPS location, and satellite images, and control the robot via the same action space, corresponding to linear and angular velocities.

Behavioral Cloning: A goal-conditioned behavioral cloning (BC) policy trained on the offline dataset that maps the three inputs to control actions [20, 99].
PPO: A policy gradient algorithm that maps the three inputs to control actions. This comparison is representative of state-of-the-art "PointGoal" navigation in simulation [119].
GCG: A model-based algorithm that uses a predictive model to plan a sequence of actions that reach the goal without causing collision [53]. We use GCG in the goal-directed mode with a GPS target, using the onboard camera and satellite images as input modalities.
RECON-H: A variant of RECON, which uses a latent goal model to represent reachable goals and plans over sampled subgoals to explore a novel environment [98]. We modify the algorithm to additionally accept the GPS and satellite images as additional inputs alongside the onboard camera image.

We evaluate the ability of ViKiNG to discover visually-indicated goals in 10 *unseen* environments of varying complexity. For each trial, we provide an RGB image of the desired target and its rough GPS location (accurate up to 5 meters). A trial is marked successful if the robot reaches the goal without requiring a human disengagement (due to a collision or getting stuck). We report the success rates of all methods in these environments in Table 5 and visualize overhead plots of the trajectories in one such environment in Figure 20.

ViKiNG outperforms all the prior methods, successfully navigating to goals that are over up to 500 meters away in our comparisons, including instances where no other method succeeds. RECON-H is the most performant of the other methods, successfully reaching most goals in the easier environments. Visualizing the robot trajectories (Fig. 20) reveals that RECON-H is unable to successfully utilize the geographic hints and explores greedily on encountering an obstacle. It also gets stuck and is unable to backtrack in 2/10 instances. While GCG also performs well in simpler environments, it is limited by its planning horizon (up to 5 seconds) and gets stuck. PPO and BC both are both unable to learn from prior data and produce collisions with bushes and a parked car, respectively. In contrast, ViKiNG is able to effectively use the local controller to avoid the obstacles and reach the goal.

Analyzing the performance in the harder tasks with ranges of up to 500 meters (Table 6), the average displacements and velocities before a user disengagement (due to collision or getting stuck) during these runs further confirm that ViKiNG is able to effectively use the geographic hints to steer the search without running into obstacles. While RECON-H manages to reach some faraway goals, it takes a greedy path to do so and is over 3× slower than ViKiNG (see Fig. 20).

4.5 THE ROLE OF GEOGRAPHIC HINTS

In this section, we closely examine the role of geographic hints on the performance of ViKiNG by studying how it deals with a low-fidelity roadmap (versus a satellite image), and with incorrect hints and degraded geographic information. For the experiment in

Method	Easy < 50m	Medium $50 - 150$m	Hard $150 - 500$m
Behavior Cloning	2/3	1/4	0/3
Offline PPO [96]	2/3	1/4	0/3
GCG [53]	**3/3**	2/4	0/3
RECON-H [98]	**3/3**	3/4	1/3
ViKiNG (Ours)	**3/3**	**4/4**	**3/3**

Table 5: Comparison of goal-seeking performance against baselines. ViKiNG successfully reaches all goals. RECON-H and GCG succeed in simpler cases but are unable to utilize the hints effectively for distant goals. PPO and BC fail in all but the simplest cases.

Method	Avg. Displacement (m)	Avg. Velocity (m/s)
Behavior Cloning	19.5	0.35
Offline PPO [96]	47.2	0.85
GCG [53]	78.3	**1.40**
RECON-H [98]	188.3	0.41
ViKiNG (Ours)	**250.0+**	**1.36**

Table 6: Average robot displacement and velocity before disengagement. ViKiNG successfully reaches all goals without requiring any disengagements. RECON-H also reaches some distant goals, but the low avg. velocity suggests that it takes an efficient path.

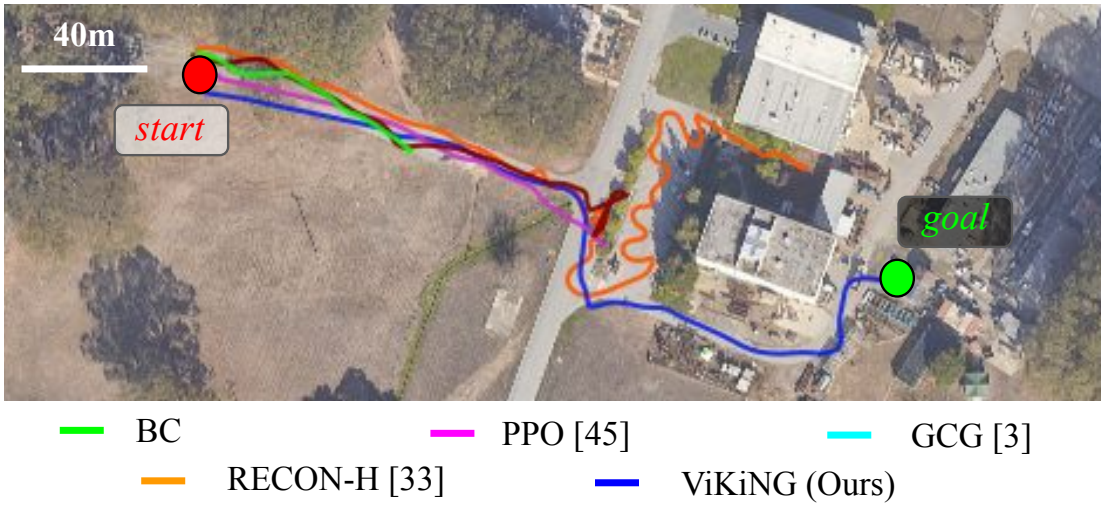


Figure 20: Trajectories taken by the methods in a *previously unseen* environment. Only ViKiNG is able to effectively use the overhead images to reach the goal (270m away) successfully, following a smooth path around the building. RECON-H and GCG get stuck, while PPO and BC result in collisions.

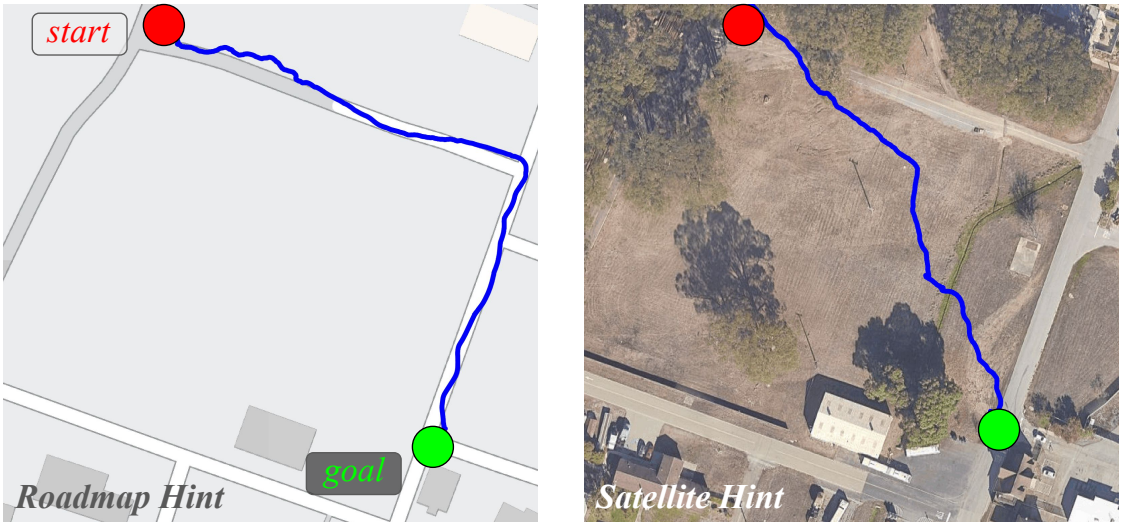


Figure 21: ViKiNG can use geographic hints in the form of a schematic roadmap or a satellite image. Providing roadmap hints encourages ViKiNG to follow marked roads (left); with satellite images, it is able to find a more direct path by cutting across a meadow (right).

Section 4.5.1, we use models trained on the same dataset, but using schematic roadmaps as geographic hints. In Sections 4.5.2 and 4.5.3, we use the same satellite image model from Section IV, with no additional retraining to accommodate missing or imperfect geographic information.

4.5.1 *Comparing Different Types of Hints*

To understand the nature of hints learned by the heuristic for different sources of geographic side information, we compare two separate versions of ViKiNG: one trained with schematic roadmaps as hints, and another trained with satellite images. Note that the method is identical in both cases, only the hint image in the data changes. For identical start-goal pairs, we observe that a model trained with roadmaps prefers following marked roads, whereas one trained with satellite images often cuts across patches of traversable terrain (e.g., grass meadows or trails) to take the quicker path, despite being trained on the same data. We hypothesize that this is due to the ability of the learned models to extract better correlations from the feature-rich satellite images, in contrast to the more abstract roadmap. Figure 21 shows a top-down view of the paths taken by the robot in the two cases in one such experiment.

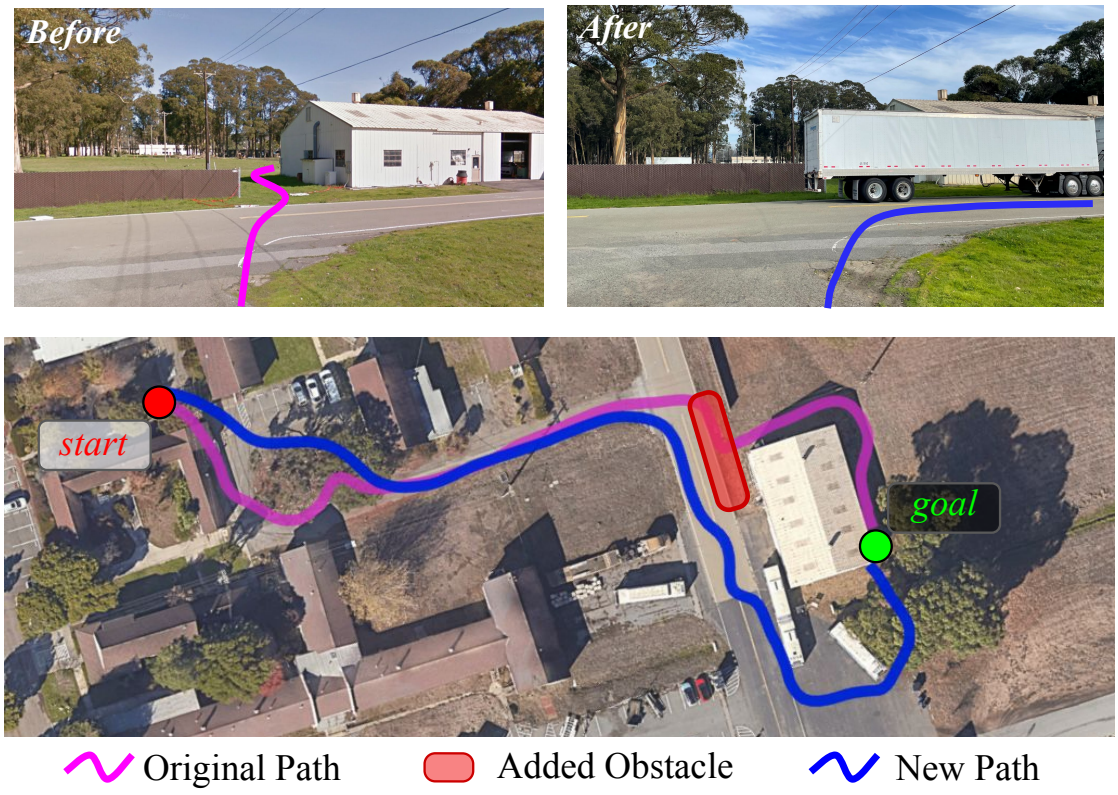


Figure 22: On navigating with outdated hints, like the truck (top right) that is absent in the satellite image, ViKiNG uses its learned local controller to propose feasible subgoals that avoid obstacles and finds a new path (blue) to the goal that avoids the truck.

4.5.2 *Outdated Hints*

To test the robustness of ViKiNG to outdated hints, we set up a goal-seeking experiment in one of the earlier environments and added a new obstacle—a large truck—blocking the path that ViKiNG took in the original trial. Since the satellite images are queried from a pre-recorded dataset, they do not reflect the addition of the truck, and hence continue to show a feasible path. We observe that the robot drives up to the truck and takes an alternate path to the goal, without colliding with it (see Figure 22). The lower-level latent goal model is robust to such obstacles and only proposes *valid* subgoal candidates that do not lead to collision; since the learned heuristic only evaluates valid subgoals, ViKiNG is robust to small discrepancies in the hints.

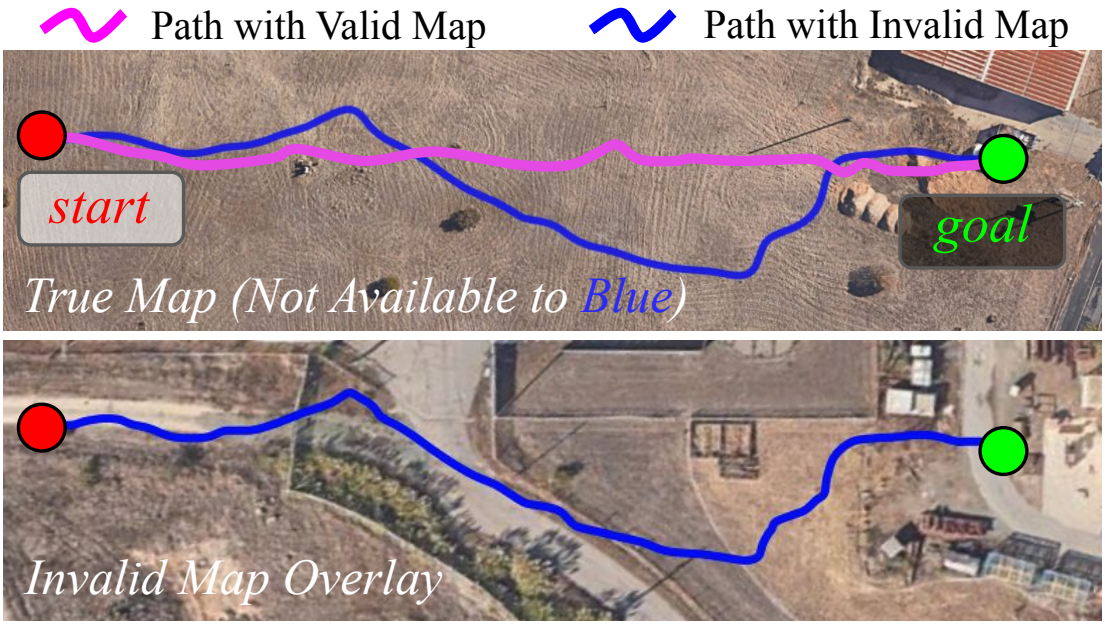


Figure 23: On navigation with invalid hints, like the map at a different location, ViKiNG deviates from its original path (magenta) and reaches the goal by following the learned heuristic (blue).

4.5.3 *Incorrect Hints*

Next, we set up a goal-seeking experiment in one of the easy environments with modified GPS measurements, so that the satellite images available to ViKiNG are offset by a ∼5km constant. As a result, this *hints* to the robot that there may be a road that it should follow, where in fact there isn't one (see Figure 23). We observe that the robot indeed deviates from its earlier path (with a valid map, the robot drives straight to the goal); upon overlaying this trajectory on the invalid map, we find that the learned heuristic indeed encourages the robot to follow the curvature of the road, but this path is still successful because it corresponds to open space.

4.5.4 *A Disoriented ViKiNG*

Finally, we analyze the effects of *disabling* the geographic hints and GPS localization on the goal-seeking performance of ViKiNG. Towards this, we run two variants of our algorithm:
No Overhead Image: We provide the robot with GPS, but no satellite images. To accommodate this, we use a simple ℓ_2 heuristic $h_{\text{GPS}}(x_w, x_g, x_t) = \|x_g - x_w\|$.
No GPS: The robot does not have access to GPS or satellite images. To accommodate this, we remove the heuristic h from ViKiNG-A*, making it an uninformed search algorithm.

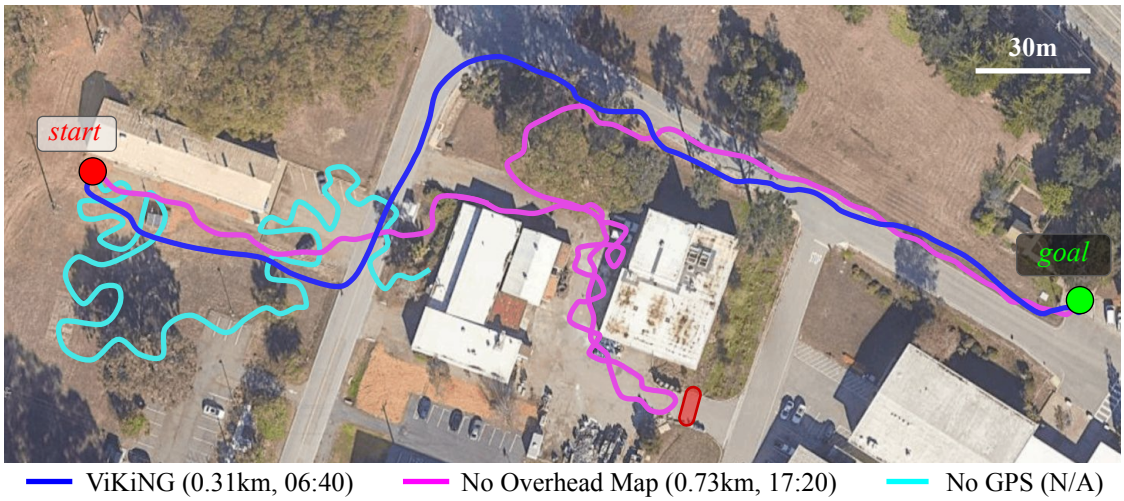


Figure 24: Ablations of ViKiNG by withholding geographic hints. ViKiNG without overhead images (magenta) acts greedily, driving close to buildings, gets caught into a cul-de-sac and eventually reaches the goal 2.6× slower that ViKiNG with access to satellite images (blue), which avoids the building cluster by following a smoother dirt path. Search without GPS (cyan) performs uninformed exploration and is unable to reach the goal in over 30 minutes.

Figure 24 summarizes the path taken by the robot, distance traversed, and time taken. When we disable the overhead hints and only use h_{GPS}, ViKiNG-A* can still reach the destination, but takes significantly longer to do so, initially exploring a dead-end path that it then has to back out of. That said, this experiment also illustrates the ability of ViKiNG-A* to handle less useful heuristics: while the path is significantly longer, the method is still able to eventually reach the destination, and in some sense the mistakes the method makes are to be expected of any system that has no prior map information. If we remove GPS as well, ViKiNG-A* corresponds to a Dijkstra-like uninformed search (resembling RECON [98]). In this case, the robot searches its environment without any guidance and is unable to reach the goal in over 30 minutes.

4.6 DISCUSSION

We proposed a method for efficiently learning vision-based navigation in *previously unseen* environments at a kilometer-scale. Our key insight is that effectively leveraging a small amount of geographic knowledge in a learning-based framework can provide strong regularities that enable robots to navigate to distant goals. We find that incorporating geographic hints as goal-directed heuristics for planning enables emergent preferences such as following roads or hiking trails. Additionally, ViKiNG only uses the hints for biasing the high-level search; the learned control policy at the lower-level relies solely on egocentric image observations, and is thus robust to imperfect hints. While we only

use overhead images in our experiments, an existing avenue for future work is to explore how such a system could use other information sources, including paper maps or textual instructions, which can be incorporated into our contrastive objective.

ACKNOWLEDGMENTS

This research was partially supported by DARPA Assured Autonomy, ARL DCIST CRA W911NF-17-2-0181, DARPA RACER, and Toyota Research Institute. The authors would like to thank Blazej Osinski, Dieter Fox, Tambet Matiisen, Brian Ichter, and Katie Kang for useful discussions.

BIBLIOGRAPHY

[1] Alessandro Achille and Stefano Soatto. "Emergence of invariance and disentanglement in deep representations". In: *The Journal of Machine Learning Research* (2018).

[2] Evan Ackerman. "Skydio demonstrates incredible obstacle-dodging full autonomy with new R1 consumer drone". In: *IEEE Spectrum* (2018).

[3] Alexander A Alemi et al. "Deep variational information bottleneck". In: *arXiv preprint arXiv:1612.00410* (2016).

[4] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. "Revisiting active perception". In: *Autonomous Robots* (2018).

[5] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst". In: *Robotics: Science and Systems (RSS)*. 2019.

[6] Marc G Bellemare et al. "Unifying count-based exploration and intrinsic motivation". In: *arXiv preprint arXiv:1606.01868* (2016).

[7] Johann Borenstein and Yoram Koren. "Real-time obstacle avoidance for fast mobile robots". In: *IEEE Transactions on systems, Man, and Cybernetics* (1989).

[8] F. Bourgault et al. "Information based adaptive robotic exploration". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2002.

[9] Guillaume Bresson et al. "Simultaneous localization and mapping: A survey of current trends in autonomous driving". In: *IEEE Transactions on Intelligent Vehicles* 2.3 (2017), pp. 194–220.

[10] Jake Bruce et al. "Learning Deployable Navigation Policies at Kilometer Scale from a Single Traversal". In: *Conference on Robot Learning (CoRL)*. 2018.

[11] Yuri Burda et al. "Exploration by random network distillation". In: *arXiv preprint arXiv:1810.12894* (2018).

[12] Cesar Cadena et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332.

[13] Devendra Singh Chaplot et al. "Learning to Explore using Active Neural SLAM". In: *International Conference on Learning Representations (ICLR)*. 2020.

[14] Devendra Singh Chaplot et al. "Semantic Curiosity for Active Visual Learning". In: *ECCV*. 2020.

[15] Benjamin Charrow et al. "Information-theoretic mapping using cauchy-schwarz quadratic mutual information". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015.

[16] Benjamin Charrow et al. "Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping". In: *Robotics: Science and Systems (RSS)*. 2015.

[17] Yevgen Chebotar et al. "Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills". In: *arXiv preprint arXiv:2104.07749* (2021).

[18] Chenyi Chen et al. "Deepdriving: Learning affordance for direct perception in autonomous driving". In: *IEEE International Conference on Computer Vision*. 2015.

[19] Hao-Tien Lewis Chiang et al. "Learning Navigation Behaviors End-to-End with AutoRL". In: *IEEE Robotics and Automation Letters* (2019).

[20] F. Codevilla et al. "End-to-End Driving Via Conditional Imitation Learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[21] Cédric Colas et al. "CURIOUS: intrinsically motivated modular multi-goal reinforcement learning". In: *International conference on machine learning*. PMLR. 2019, pp. 1331–1340.

[22] C. Connolly. "The determination of next best views". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. 1985.

[23] FR Dalgleish, SW Tetlow, and RL Allwood. "Vision-based navigation of unmanned underwater vehicles: a survey. Part I: Vision Based Cable-, Pipeline-and Fish Tracking". In: *Journal of Marine Design and Operations*. 7. 2004, pp. 51–56.

[24] DARPA. *Subterranean Challenge*. 2019. URL: https://www.subtchallenge.com.

[25] Andrew J Davison and David W Murray. "Mobile robot localisation using active vision". In: *European conference on computer vision*. Springer. 1998, pp. 809–825.

[26] Jeffrey Delmerico et al. "Active Autonomous Aerial Exploration for Ground Robot Path Planning". In: *IEEE Robotics and Automation Letters* (2017).

[27] Yiming Ding et al. "Goal-conditioned Imitation Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[28] Yiming Ding et al. "Goal-conditioned imitation learning". In: *arXiv preprint arXiv:1906.05838* (2019).

[29] Yan Duan et al. "RL^2: Fast reinforcement learning via slow reinforcement learning". In: *arXiv preprint arXiv:1611.02779* (2016).

[30] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. "Challenges of real-world reinforcement learning". In: *arXiv preprint arXiv:1904.12901* (2019).

[31] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. "Search on the Replay Buffer: Bridging Planning and RL". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[32] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. "C-Learning: Learning to Achieve Goals via Recursive Classification". In: *arXiv preprint arXiv:2011.08909* (2020).

[33] A. Faust et al. "PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[34] Aleksandra Faust et al. "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.

[35] Patrick S. Foo et al. "Do humans integrate routes into a cognitive map? Map- versus landmark-based navigation of novel shortcuts." In: *Journal of experimental psychology. Learning, memory, and cognition* (2005).

[36] Patrick S. Foo et al. "Do humans integrate routes into a cognitive map? Map- versus landmark-based navigation of novel shortcuts." In: *Journal of experimental psychology. Learning, memory, and cognition* (2005).

[37] A. Francis et al. "Long-Range Indoor Navigation With PRM-RL". In: *IEEE Transactions on Robotics* (2020).

[38] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. "Visual simultaneous localization and mapping: a survey". In: *Artificial Intelligence Review* (2015).

[39] Paul Furgale and Timothy D Barfoot. "Visual teach and repeat for long-range rover autonomy". In: *Journal of Field Robotics* (2010).

[40] Shani Gamrian and Yoav Goldberg. "Transfer learning for related reinforcement learning tasks via image-to-image translation". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2063–2072.

[41] Dibya Ghosh et al. "Learning to reach goals without reinforcement learning". In: *arXiv preprint arXiv:1912.06088* (2019).

[42] S. Gillner and H. A. Mallot. "Navigation and Acquisition of Spatial Knowledge in a Virtual Maze". In: *Journal of Cognitive Neuroscience* (1998).

[43] Anirudh Goyal et al. "Infobot: Transfer and exploration via the information bottleneck". In: *arXiv preprint arXiv:1901.10902* (2019).

[44] Abhishek Gupta et al. "Learning invariant feature spaces to transfer skills with reinforcement learning". In: *arXiv preprint arXiv:1703.02949* (2017).

[45] Kristian Hartikainen et al. "Dynamical Distance Learning for Semi-Supervised and Unsupervised Skill Discovery". In: *International Conference on Learning Representations*. 2020.

[46] Elad Hazan et al. "Provably efficient maximum entropy exploration". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2681–2691.

[47] Dirk Holz et al. *A comparative evaluation of exploration strategies and heuristics to improve them*. 2011.

[48] Rein Houthooft et al. "Vime: Variational information maximizing exploration". In: *arXiv preprint arXiv:1605.09674* (2016).

[49] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[50] Maximilian Igl et al. "Generalization in reinforcement learning with selective noise injection and information bottleneck". In: *arXiv preprint arXiv:1910.12911* (2019).

[51] Krishna Murthy Jatavallabhula, Ganesh Iyer, and Liam Paull. "gradSLAM: Dense SLAM meets Automatic Differentiation". In: *CoRR* (2019).

[52] Leslie Pack Kaelbling. "Learning to achieve goals". In: *IJCAI*. Citeseer. 1993, pp. 1094–1099.

[53] G. Kahn et al. "Self-Supervised Deep RL with Generalized Computation Graphs for Robot Navigation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[54] Gregory Kahn, Pieter Abbeel, and Sergey Levine. "BADGR: An Autonomous Self-Supervised Learning-Based Navigation System". In: *IEEE Robotics and Automation Letters* (2021).

[55] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[56] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)* (2015).

[57] Thomas Kollar and Nicholas Roy. "Efficient Optimization of Information-Theoretic Exploration in SLAM". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2008.

[58] Eric Kolve et al. "AI2-THOR: An Interactive 3D Environment for Visual AI". In: *ArXiv* (2019).

[59] Eric Krotkov and Martial Hebert. "Mapping and positioning for a prototype lunar rover". In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. IEEE. 1995.

[60] Benjamin Kuipers and Yung-Tai Byun. "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations". In: *Robotics and Autonomous Systems* (1991). Special Issue Toward Learning Robots.

[61] Ashish Kumar* et al. "Visual Memory for Robust Path Following". In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[62] T.L Lai and Herbert Robbins. "Asymptotically efficient adaptive allocation rules". In: *Advances in Applied Mathematics* (1985).

[63] Steven M LaValle. *Planning Algorithms*. Cambridge university press, 2006.

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

[65] Lisa Lee et al. "Efficient exploration via state marginal matching". In: *arXiv preprint arXiv:1906.05274* (2019).

[66] S. Levine and D. Shah. "Learning robotic navigation from experience: principles, methods and recent results". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* (2023).

[67] Sergey Levine et al. "End-to-End Training of Deep Visuomotor Policies". In: *The Journal of Machine Learning Research* (2016).

[68] Sergey Levine et al. *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. 2020. arXiv: 2005.01643 [cs.LG].

[69] Kara Liu et al. "Hallucinative topological memory for zero-shot visual planning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6259–6270.

[70] Iker Lluvia, Elena Lazkano, and Ander Ansuategi. "Active Mapping and Robot Exploration: A Survey". In: *Sensors* (2021).

[71] Corey Lynch et al. "Learning latent plans from play". In: *Conference on Robot Learning*. PMLR. 2020, pp. 1113–1132.

[72] Manolis Savva* et al. "Habitat: A Platform for Embodied AI Research". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[73] John McCormac et al. "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[74] M. Meng and A. C. Kak. "Mobile Robot Navigation using Neural Networks and Nonmetrical Environmental Models". In: *IEEE Control Systems Magazine* (1993).

[75] M. Meng and A. C. Kak. "NEURO-NAV: A Neural Network based Architecture for Vision-guided Mobile Robot Navigation using Non-metrical Models of the Environment". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1993.

[76] X. Meng et al. "Scaling Local Control to Large-Scale Topological Navigation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[77] Atanas Mirchev et al. "Approximate bayesian inference in spatial environments". In: *arXiv preprint arXiv:1805.07206* (2018).

[78] Anusha Nagabandi et al. "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning". In: *arXiv preprint arXiv:1803.11347* (2018).

[79] Medhini Narasimhan et al. "Seeing the Un-Scene: Learning Amodal Semantic Maps for Room Navigation". In: *CoRR* (2020).

[80] John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*. Oxford: Clarendon Press, 1978. ISBN: 0-19-857206-9.

[81] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2019.

[82] Christos Papachristos, Shehryar Khattak, and Kostas Alexis. "Uncertainty-aware receding horizon exploration and mapping using aerial robots". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[83] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. "Actor-mimic: Deep multitask and transfer reinforcement learning". In: *arXiv preprint arXiv:1511.06342* (2015).

[84] Deepak Pathak et al. "Curiosity-driven exploration by self-supervised prediction". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2778–2787.

[85] Silviu Pitis et al. "Maximum entropy gain exploration for long horizon multi-goal reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7750–7761.

[86] Vitchyr Pong et al. "Temporal difference models: Model-free deep rl for model-based control". In: *arXiv preprint arXiv:1802.09081* (2018).

[87] Vitchyr H Pong et al. "Skew-fit: State-covering self-supervised reinforcement learning". In: *arXiv preprint arXiv:1903.03698* (2019).

[88] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. "Deep Imitative Models for Flexible Inference, Planning, and Control". In: *International Conference on Learning Representations*. 2020.

[89] C. Rosen and N. Nilsson. "APPLICATION OF INTELLIGENT AUTOMATA TO RECONNAISSANCE." In: *SRI Technical Report*. 1967.

[90] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[91] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. "Meta reinforcement learning with latent variable gaussian processes". In: *arXiv preprint arXiv:1803.07551* (2018).

[92] Yash Satsangi et al. "Maximizing Information Gain in Partially Observable Environments via Prediction Reward". In: *arXiv preprint arXiv:2005.04912* (2020).

[93] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. "Semi-Parametric Topological Memory for Navigation". In: *International Conference on Learning Representations*. 2018.

[94] Nikolay Savinov et al. "Episodic curiosity through reachability". In: *International Conference on Learning Representations (ICLR* (2019).

[95] Tom Schaul et al. "Universal Value Function Approximators". In: *International Conference on Machine Learning (ICML)*. 2015.

[96] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[97] Magnus Selin et al. "Efficient Autonomous Exploration Planning of Large-Scale 3-D Environments". In: *IEEE Robotics and Automation Letters* (2019).

[98] Dhruv Shah et al. "Rapid Exploration for Open-World Navigation with Latent Goal Models". In: *5th Annual Conference on Robot Learning*. 2021.

[99] Dhruv Shah et al. "ViNG: Learning Open-World Navigation with Visual Goals". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[100] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer Handbook of Robotics*. Vol. 200. Springer, 2008.

[101] R. Sim and J. J. Little. "Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2006.

[102] D. Singh Chaplot et al. "Neural Topological SLAM for Visual Navigation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[103] Rupesh Kumar Srivastava et al. "Training agents using upside-down reinforcement learning". In: *arXiv preprint arXiv:1912.02877* (2019).

[104] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. "Incentivizing exploration in reinforcement learning with deep predictive models". In: *arXiv preprint arXiv:1507.00814* (2015).

[105] Hao Sun et al. "Policy continuation with hindsight inverse dynamics". In: *arXiv preprint arXiv:1910.14055* (2019).

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

[107] Richard S. Sutton. "Learning to predict by the methods of temporal differences". In: *Machine Learning* (1988).

[108] Wennie Tabib et al. "Real-Time Information-Theoretic Exploration with Gaussian Mixture Model Maps." In: *Robotics: Science and Systems*. 2019.

[109] Matthew E Taylor and Peter Stone. "Cross-domain transfer for reinforcement learning". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 879–886.

[110] Charles Thorpe et al. "Vision and navigation for the Carnegie-Mellon Navlab". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1988).

[111] Sebastian Thrun. "Simultaneous localization and mapping". In: *Robotics and cognitive approaches to spatial mapping*. Springer, 2007, pp. 13–41.

[112] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005. ISBN: 0262201623.

[113] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. "Probalistic robotics". In: *Kybernetes* (2006).

[114] Sebastian Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge". In: *Journal of field Robotics* (2006).

[115] Naftali Tishby, Fernando C Pereira, and William Bialek. "The information bottleneck method". In: *arXiv preprint physics/0004057* (2000).

[116] Chris Urmson et al. "Autonomous driving in urban environments: Boss and the urban challenge". In: *Journal of Field Robotics* (2008).

[117] Yan Wang et al. "Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[118] Jan M. Wiener, Simon J. Büchner, and Christoph Hölscher. "Taxonomy of Human Wayfinding Tasks: A Knowledge-Based Approach". In: *Spatial Cognition & Computation* (2009).

[119] Erik Wijmans et al. "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames". In: *International Conference on Learning Representations (ICLR)*. 2020.

[120] Huazhe Xu et al. "End-to-end learning of driving models from large-scale video datasets". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2174–2182.

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

[122] Naoki Yokoyama, Sehoon Ha, and Dhruv Batra. *Success Weighted by Completion Time: A Dynamics-Aware Evaluation Criteria for Embodied Navigation*. 2021. arXiv: 2103.08022 [cs.RO].

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

[124] Y. Zhu et al. "Target-driven visual navigation in indoor scenes using deep reinforcement learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

APPENDICES

APPENDIX A: OPEN-WORLD EXPLORATION WITH LATENT GOAL MODELS

A.1 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, including parts collected by Kahn et al. [54] and Shah et al. [99] for earlier research projects, and exhibits significant variation in appearance due to seasonal and lighting changes.

This dataset is available for download at sites.google.com/view/recon-robot/dataset, along with helper scripts to load and visualize the trajectories.

A.1.1 *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 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 [54]. 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.1.2 *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 25).

Figure 26 shows the exploration and navigation performance of RECON and the baselines (see Sec. 3.5 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.

A.2 REPRODUCIBILITY

A.2.1 *Algorithmic Components*

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 $q_\theta(a_t, d_t | z_t^w, o_\tau)$ 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 AssociateToVertex subroutine. To nudge the robot to the frontier, we use a heuristic 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 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. 6.

Algorithm 6 Pseudocode for subroutines referenced in the exploration algorithm shown in Alg. 3

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*
    end for
    $v_H \leftarrow$ 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 $\mathcal{D}_w, o_H$
end function

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*
    $v, d \leftarrow \mathbf{d}$[o] ▷ []*Associate o_t with nearest vertex*
    return v
end function

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
end function

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*
end procedure

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

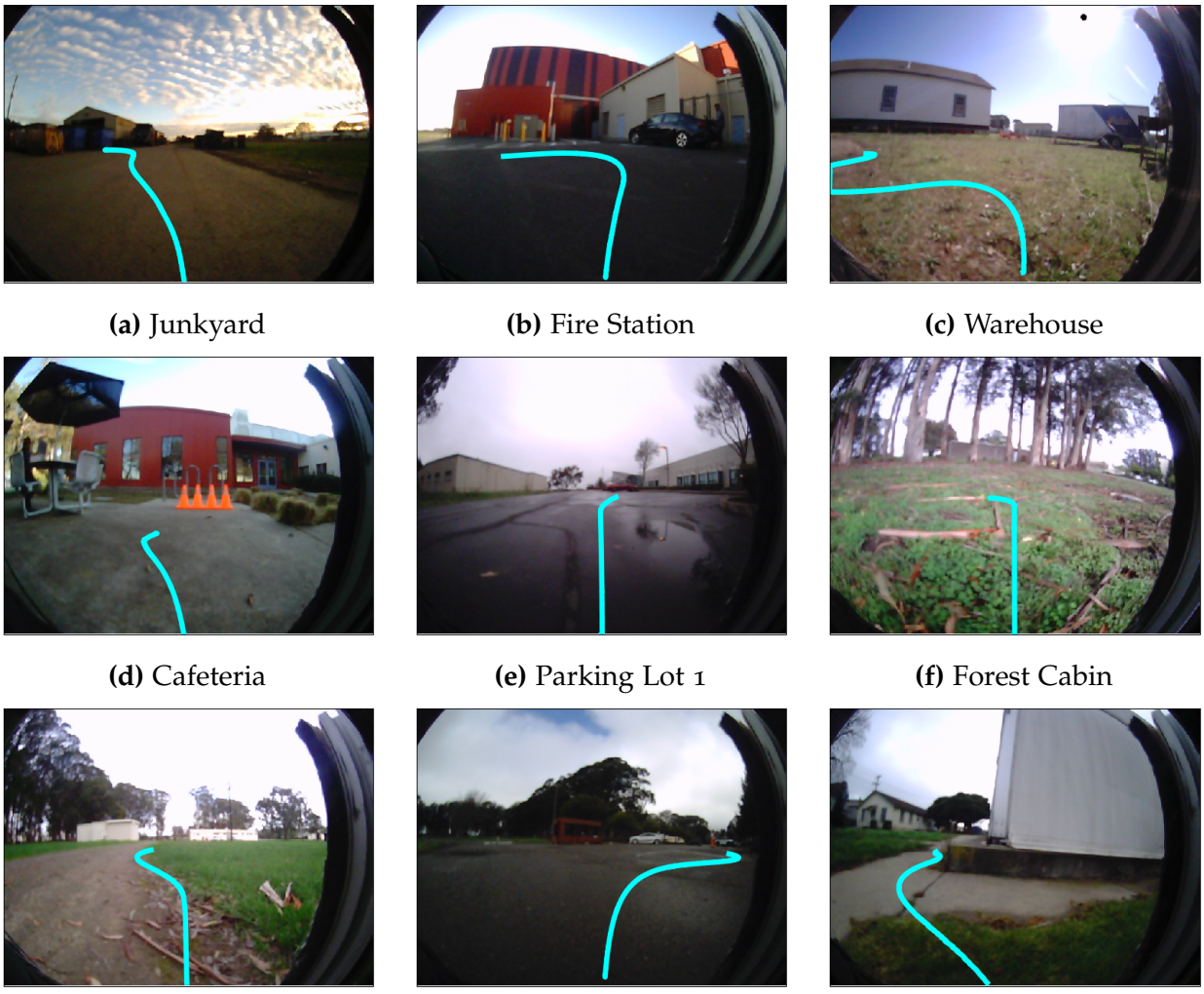
Table 7: Hyperparameters used in our experiments.

A.2.2 *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 [49] followed by a fully-connected layer projecting the 1024-dimensional latents to a stochastic, context-conditioned representation 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 [55] 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 8. 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 7.

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 dimension.
2	I_t^g [6, 160, 120]	E_t^g [1024]	MobileNet Encoder [49]
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 [49]
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]	$\vec{a}_t^g$ [2], $\vec{d}_t^g$ [1]	Split into actions and distances.

Table 8: 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.



(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 25: We collect data in 9 diverse environments. Example trajectories are shown in cyan.

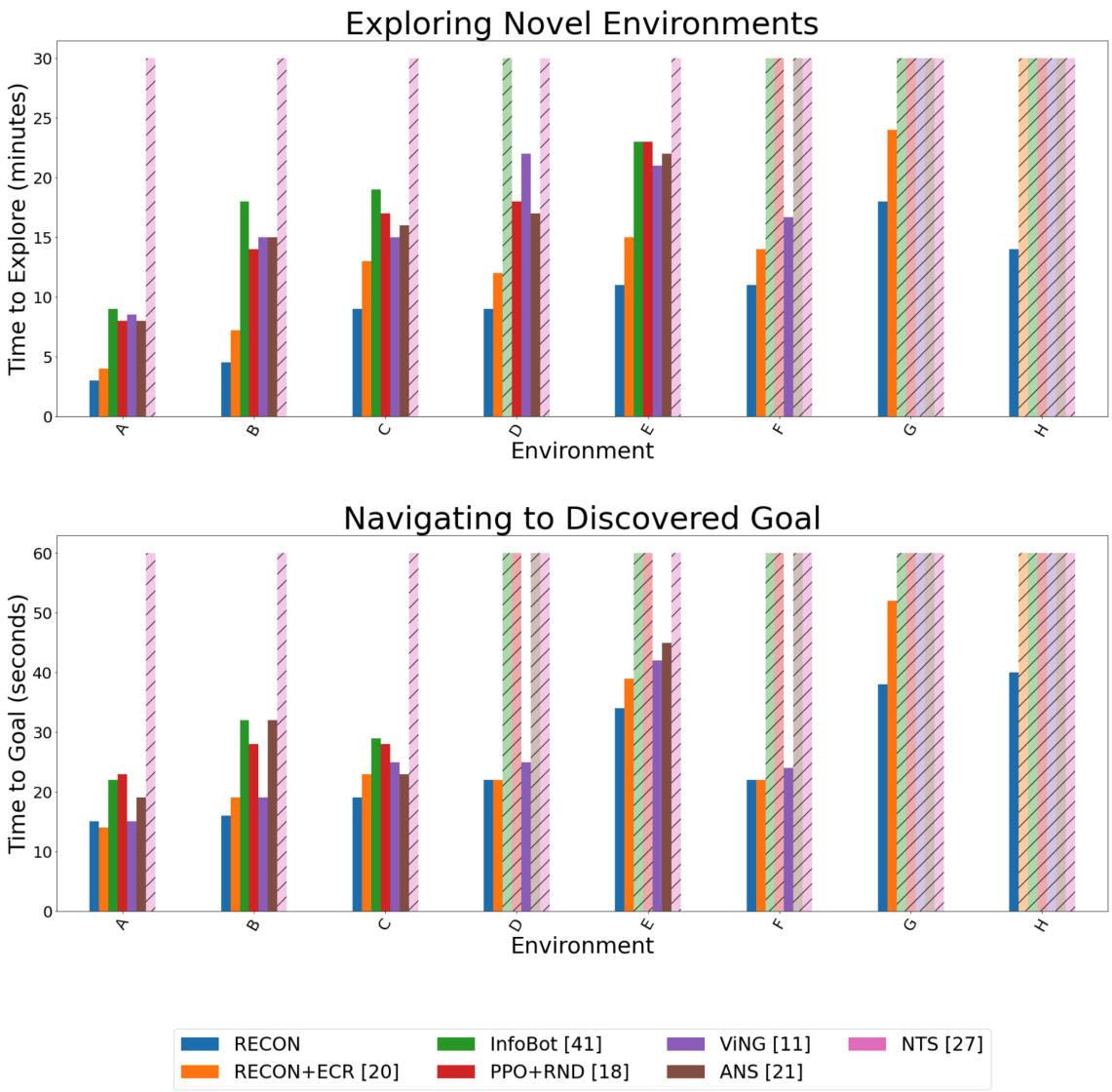


Figure 26: Exploring and learning to reach goals: *(left)* Amount of time needed for each method to search for the goals in a new environment (↓ is better; hashed out bars represent failure). *(right)* Amount of time needed to reach the goal a second time, after reaching the goal once and constructing the map, in seconds (↓ is better).

APPENDIX B: KILOMETER-SCALE EXPLORATION WITH GEOGRAPHIC HINTS

B.1 IMPLEMENTATION DETAILS

Layer	Input [Dimensions]	Output [Dimensions]	Layer Details
		Encoder $p_\phi(z \mid o_t, o_w) = \mathcal{N}(\cdot; \mu_p, \Sigma_p)$	
1	o_t, o_w [3, 160, 120]	I_t^w [6, 160, 120]	Concatenate along channel dimension.
2	I_t^w [6, 160, 120]	E_t^w [1024]	MobileNet Encoder [49]
3	E_t^w [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 $q_\theta(a, d, x \mid o_t, z_t^w) = \mathcal{N}(\cdot; \mu_q, \Sigma_q)$	
1	o_t [3, 160, 120]	E_t [1024]	MobileNet Encoder [49]
2	E_t [1024], z_t^w [64]	$F = E_t \oplus z_t^w$ [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 [5]	Σ_q [5, 5]	torch.diag(σ_q)
5	μ_q [5]	$\bar{a}_t^w$[2], $\bar{d}_t^w$[1], $\bar{x}_t^w$[2]	Split into actions, distances and offsets

Table 9: Architectural details of the latent goal model (Section 4.3.1)

B.1.1 *Latent Goal Model (Section 4.3.1)*

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 [49] followed by a fully-connected layer projecting the 1024-dimensional latents to a stochastic, context-conditioned representation z_t^w 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^w. We use the reparametrization trick [55] to sample from the latent and use the concatenated encodings to learn the optimal actions a_t^w, temporal distances d_t^w and spatial offsets x_t^w. Details of our network architecture are provided in Table 9. During pretraining, we maximize $\mathcal{L}_{\text{VIB}}$ (Eq. 3) with a batch size of 128 and perform gradient updates using the Adam optimizer with learning rate $\lambda = 10^{-4}$ until convergence.

B.1.2 *Learned Heuristic (Section 4.3.3)*

Inputs to the encoder p_{over} are (i) satellite image c_S and (ii) the triplet of GPS locations $\{x_w, x_S, x_G\}$. p_{over} is implemented as a multi-input neural network with a MobileNet encoder [49] to featurize c_S, which is then concatenated with the location inputs. This is followed by a series of fully-connected layers [512, 128, 32, 1] down to a single cell to predict the binary classification scores. During pretraining, we minimize $\mathcal{L}_{\text{NCE}}$ with a batch size of 256 and perform gradient updates using the Adam optimizer with learning rate $\lambda = 10^{-4}$ until convergence.

B.1.3 *Miscellaneous Hyperparameters*

We provide the hyperparameters associated with our algorithms in Table 10.

Hyperparameter	Value	Meaning
Δt	0.5	Time step of the robot (s)
ϵ	10	Threshold for close (Sec. 4.3.2)
C	20	Scaling constant for v (Alg. 5 L15)
λ_{over}	200	Scaling constant for h_{over} (Sec. 4.3.3)

Table 10: Hyperparameters used in our experiments.

B.2 OFFLINE TRAJECTORY DATASET

For the offline dataset discussed in Section 4.4.2, we use a combination of a 30 hours of autonomously collected data, and 12 hours of human teleoperated data. The complete dataset was collected by 3 independent sets of researchers over the course of 24 months in environments spanning multiple cities. We provide more information below.

B.2.1 *Autonomously Collected Data*

We use the published dataset by Shah et al. [98], that contains 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.

During data collection, a robot is equipped with a 2D LIDAR sensor to detect collisions ahead of time and generate autonomous pseudo-labels for collision events. To ensure that the control policy achieves sufficient coverage of the environment while also ensuring

	Training Dataset	ViKiNG Deployment
Avg. Length	45m	>1km
Avg. Velocity (m/s)	1.68	1.36

Table 11: Trajectory statistics for offline training dataset and real-world deployment.

Environment Type	Amount of Data (hrs)
Paved Hiking Trails	01:45
City Sidewalks	02:15
Suburban Neighborhood Roads	01:30
Unpaved Grasslands	01:00
University/Office Campus	02:30
Miscellaneous	03:00
Total	**12:00**

Table 12: Approximate composition of various environment types in the teleoperated dataset.

that the action sequences executed by the robot are realistic, we use a time-correlated random walk to gather data.

B.2.2 *Human Teleoperated Data*

The above dataset contains extremely diverse dataset that is great for learning general notions of traversability and collision avoidance. However, the random nature of the dataset means that it does not contain any semantically interesting behavior that may be desired of a robotic system, such as following a sidewalk or through a patch of trees. To enhance the quality of learned behaviors, we augment this dataset with about 12 hours of human teleoperated data in semantically rich environments such as hiking trails, city sidewalks, parking lots and suburban neighborhoods. These environments represent realistic scenarios where such a robotic system would be deployed.

Table 11 summarizes key statistics of the trajectories, such as length and velocity. Table 12 summarizes the various environments in which the dataset was collected, and their relative composition. Figure 27 visualizes the geographic locations of these data collection sites (location anonymized for the double-blind review process). We ensure no overlap between the training and test environments—success in these test environments requires *true* generalization to unseen environments.

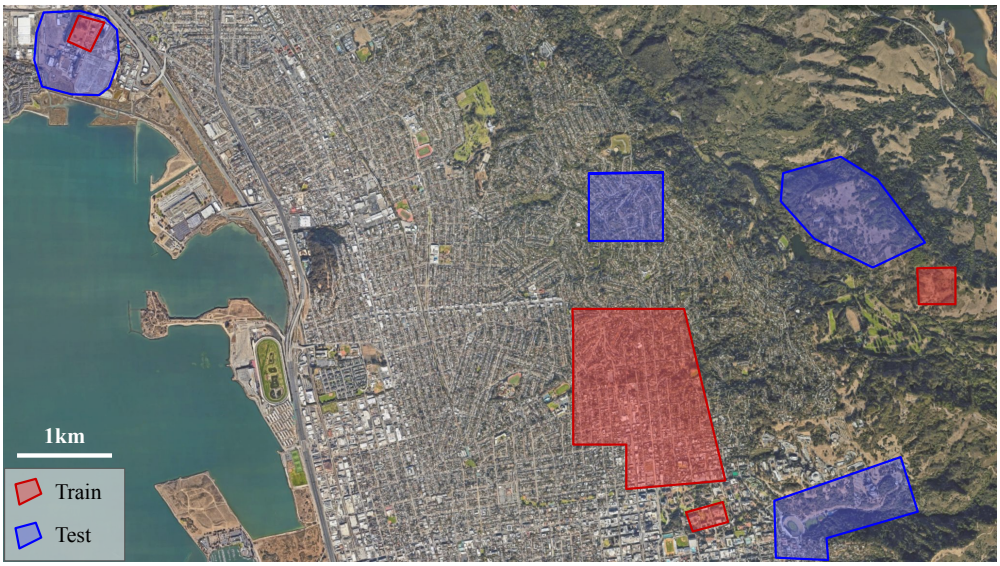


Figure 27: Rough geographical locations of data collection by human teleoperation and testing (Section 4.4)