

ViNG: Learning Open-World Navigation with Visual Goals

Dhruv Shah¹, Benjamin Eysenbach², Gregory Kahn¹, Nicholas Rhinehart¹, Sergey Levine¹

¹UC Berkeley, ²Carnegie Mellon University

Abstract—We propose a learning-based navigation system for reaching visually indicated goals and demonstrate this system on a real mobile robot platform. Learning provides an appealing alternative to conventional methods for robotic navigation: instead of reasoning about environments in terms of geometry and maps, learning can enable a robot to learn about navigational affordances, understand what types of obstacles are traversable (e.g., tall grass) or not (e.g., walls), and generalize over patterns in the environment. However, unlike conventional planning algorithms, it is harder to change the goal for a learned policy during deployment. We propose a method for learning to navigate towards a goal image of the desired destination. 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, including other methods that incorporate reinforcement learning and search. 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. We encourage the reader to visit the project website for videos of our experiments and demonstrations¹.

I. 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 [1–3], and rely heavily on learning. Further, in the absence of spatial positional information (e.g., GPS or maps), specification of a navigational goal itself becomes challenging, since locational goals require the robot to be able to compare its location to the target.

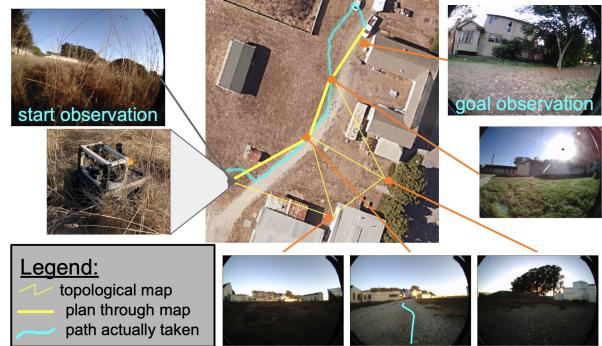


Fig. 1: ViNG builds and plans over a *learned* topological graph consisting of previously seen egocentric images, and uses a learned controller to execute the path to a visually indicated goal. Unlike prior work, our method uses purely offline experience and does not require a simulator or online data 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.

In this paper, 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 – visual navigation with goals. 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,

¹Project website: sites.google.com/view/ving-robot

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.

II. RELATED WORK

Prior work has studied vision-based mobile robot navigation in many real-world settings, including indoor and outdoor navigation [4–6], autonomous driving [7, 8], and navigation in extra-terrestrial and underwater environments [9, 10]. The combination of mapping [11] and path planning [12] has been a cornerstone for a number of effective systems [13–15] and underlies several state-of-the-art navigation systems [16, 17]. 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 [18]. Further, prior work often assumes that geometric traversability is faithfully indicated through observations and not misled by (say) non-obstacles such as tall grass [19]. Learning-based systems lift some of these assumptions and can use learned models to perform perception [20, 21], planning [22–24], or both [25]. In practice, learning temporally extended long-horizon skills with either reinforcement learning (RL) or imitation learning (IL) remains difficult [26, 27].

Recent methods address limitations of the above approaches by combining planning and learning [28–33]. These methods use learning (i.e., approximate dynamic programming) to solve short-horizon tasks and plan (i.e., use exact dynamic programming) over non-metric topological graphs [34, 35] 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



Fig. 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.

to an exact simulation replica of the environment [33, 36], assuming simplified action spaces [28–30], or requiring online data collection [28, 30]. Our experiments in Section V 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.

III. 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, \dots, 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.

A. 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.

B. 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 [37]; 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.

IV. 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 *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 [28, 29, 33], our experiments demonstrate that two key technical insights contribute to significantly improved performance in the real-world setting: *graph pruning* (Sec. IV-B2) and *negative mining* (Sec. IV-A1). Our comparisons to prior methods in Section V and ablation studies in Section V-D 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.

Algorithm 1 Training ViNG

```

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

```

A. 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 [38, 39]. 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 [39]. 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 II), 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.

1) *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, possibly 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

Algorithm 2 Deploying ViNG

- 1: **Input** current image o_t , goal image o_G , and topological graph \mathcal{M} .
 - 2: Add o_t, o_G to the map \mathcal{M} using distances from \mathcal{T} .
 - 3: $o_{w_1}, o_{w_2}, \dots \leftarrow \text{Dijkstra}(\text{start} = o_t, \text{goal} = o_G, \mathcal{M})$
 - 4: Estimate relative pose of first waypoint: $\Delta p \leftarrow \mathcal{P}(o_t, o_{w_1})$
 - 5: $u_t \leftarrow \text{PD-CONTROLLER}(\Delta p)$
 - 6: **return** control u_t
-

improvements over prior methods.

B. 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} .

1) *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 ($\mathcal{T}(o_i, o_j) < \delta_{\text{sparsify}}$) are not added to the graph, since the controller can traverse those edges with high probability.

2) *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.

C. 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 V-D.

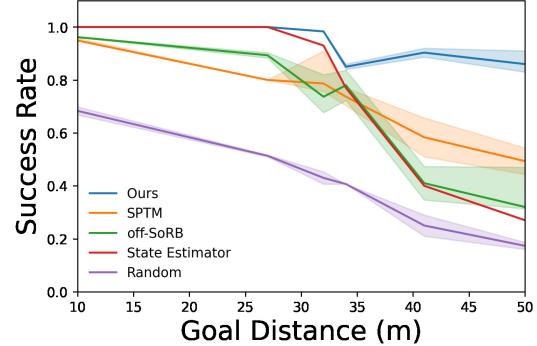


Fig. 3: Real-World Navigation: While all non-random methods successfully reach nearby goals, only ViNG reaches goals over 40 meters away. Here, success rate is defined as the average over portion of the expert trajectory to goal that each run successfully completes.

D. 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 [40] 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 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 $\{1, \dots, d_{\max} = 50\}$ 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 [41] with learning rate $\lambda = 10^{-4}$. Algorithms 1 and 2 summarize our approach in the training and deployment stages, respectively.

V. 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 IV-C?

A. 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 [37] 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 [29]

off-SoRB: an offline variant of SoRB that uses a topological graph and offline RL to learn a distributional Q-function [28]

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 IV-C

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* [33] uses a photorealistic simulator to train its distance and action models, using $\sim 1.5\text{M}$ samples, while *PRM-RL* [36] 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 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 IV. We visualize trajectories in Fig. 4.

B. 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



Fig. 4: Qualitative Results in the urban Environment: Each approach was directed to a visual goal $\sim 50\text{m}$ 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.

in Fig. 5, all of which are distinct from the setting used in our main experiments (Sec. V-A). In each new environment, a human controlled the robot to provide initial exploration data. After this initial data collection, the robot collected experience *autonomously*: it randomly sampled a previously-observed image as the goal 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. V-A, 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 I 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. V-A), 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. V-A 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.

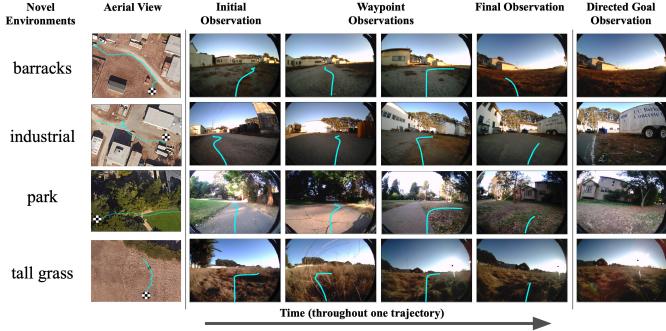


Fig. 5: Generalization Experiments: We evaluate ViNG in four new outdoor environments. For each, we collect a few dozen minutes of experience to adapt the distance function and relative pose predictor. Then, given a goal image (last column, checkerboard location in aerial view), the robot attempts to 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.

| 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 I: 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.

C. Comparisons to Online Methods

While Section V-A 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 a Unity-based, photorealistic outdoor navigation simulator. We include new additional

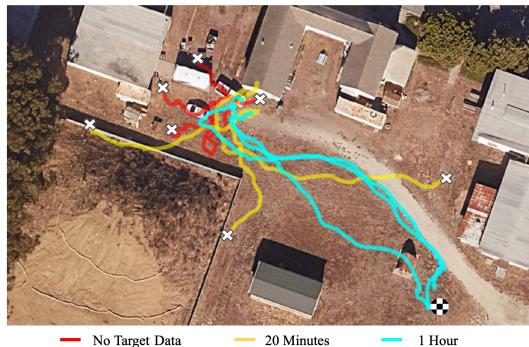


Fig. 6: Fast Adaptation to a New Environment: After training ViNG in one environment, we deploy the system in a novel environment, shown above. By practicing to reach self-proposed goals and using that experience to finetune the controller, ViNG is able to quickly gain competence at reaching distance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

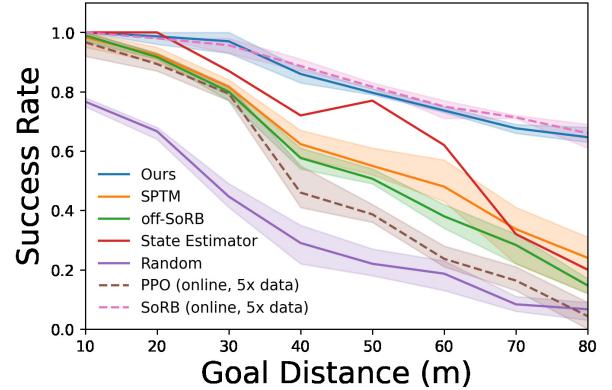


Fig. 7: Results from Simulated Navigation: ViNG is substantially more successful at reaching distance goals than all offline baselines, while performing competitively with SoRB, a popular online baseline combining Q-learning and topological graphs. We emphasize that SoRB and PPO require 5× online data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

baselines in the simulated experiment:

PPO: a popular reactive controller for indoor visual navigation algorithms [42, 43]

SoRB: online version of the “off-SoRB” baseline [28]

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.

D. Ablation Experiments

A key design decision for ViNG that differentiates it from prior methods (e.g., [28, 33]) 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 II. Two simple baselines, “direct actions” and “direct actions (discrete)”, use the goal-conditioned behavior cloning method of [29, 44] 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.

E. Applications and Qualitative Results

ViNG’s ability to navigate using perception and landmarks, without access to maps or localization, can enable a number of

| 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 II: 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.

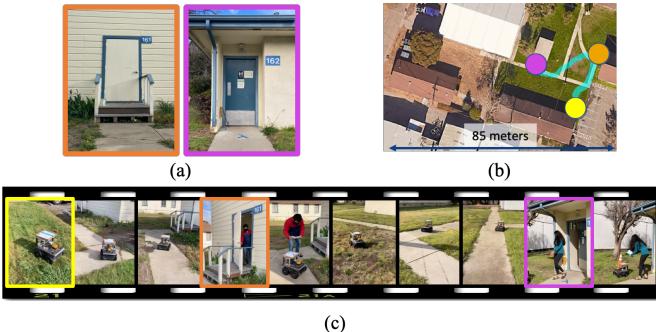


Fig. 8: Contactless Last-Mile Delivery Demo: Given a set of visually-indicated goals (a), ViNG can perform contactless delivery in the urban neighborhood successfully, as shown in the filmstrip (c). An overhead view (b) with starting position marked in yellow and respective goals marked in orange and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

intuitive applications, which we illustrate through 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).

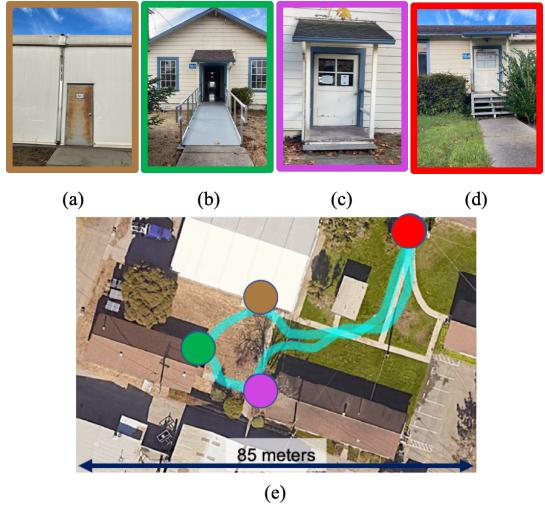


Fig. 9: Autonomous Inspection Demo: Given a set of visual landmarks (a–d) in a university campus, ViNG can perform autonomous inspection by navigating to these goals periodically. An overhead view (b) shows color-coded goals and the trajectory taken by robot (cyan) in one cycle. Note: The satellite view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

VI. CONCLUSION

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 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.

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.

REFERENCES

- [1] J. O’Keefe and L. Nadel, *The Hippocampus as a Cognitive Map*. Oxford: Clarendon Press, 1978.
- [2] S. Gillner and H. A. Mallot, “Navigation and acquisition of spatial knowledge in a virtual maze,” *Journal of Cognitive Neuroscience*, 1998.
- [3] P. S. Foo, W. Warren, A. Duchon, and M. Tarr, “Do humans integrate routes into a cognitive map? map- versus landmark-based navigation of novel shortcuts.” *Journal of experimental psychology. Learning, memory, and cognition*, 2005.
- [4] C. Rosen and N. Nilsson, “Application of intelligent automata to reconnaissance.” in *SRI Technical Report*, 1967.
- [5] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer, “Vision and navigation for the carnegie-mellon navlab,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, 1988.
- [6] J. Borenstein and Y. Koren, “Real-time obstacle avoidance for fast mobile robots,” *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.
- [7] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, “Stanley: The robot that won the darpa grand challenge,” *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [8] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, “Autonomous driving in urban environments: Boss and the urban challenge,” *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
- [9] E. Krotkov and M. Hebert, “Mapping and positioning for a prototype lunar rover,” in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 1995.
- [10] F. Dalgleish, S. Tetlow, and R. 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*, no. 7, 2004, pp. 51–56.
- [11] S. Thrun, W. Burgard, and D. Fox, “Probabilistic robotics,” *Kybernetes*, 2006.
- [12] S. M. LaValle, *Planning Algorithms*. Cambridge university press, 2006.
- [13] A. J. Davison and D. W. Murray, “Mobile robot localisation using active vision,” in *European conference on computer vision*. Springer, 1998, pp. 809–825.
- [14] 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*, 2006.
- [15] P. Furgale and T. D. Barfoot, “Visual teach and repeat for long-range rover autonomy,” *Journal of Field Robotics*, 2010.
- [16] M. Bansal, A. Krizhevsky, and A. Ogale, “ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst,” in *Robotics: Science and Systems (RSS)*, 2019.
- [17] E. Ackerman, “Skydio demonstrates incredible obstacle-dodging full autonomy with new r1 consumer drone,” *IEEE Spectrum*, 2018.
- [18] DARPA. (2019) Subterranean Challenge. [Online]. Available: <https://www.subtchallenge.com>
- [19] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” *Artificial Intelligence Review*, 2015.
- [20] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving,” in *IEEE International Conference on Computer Vision*, 2015.
- [21] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, “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.
- [22] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, “Self-Supervised Deep RL with Generalized Computation Graphs for Robot Navigation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [23] A. Kumar*, S. Gupta*, D. Fouhey, S. Levine, and J. Malik, “Visual Memory for Robust Path Following,” in *Neural Information Processing Systems (NeurIPS)*, 2018.
- [24] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine, “Dynamical Distance Learning for Semi-Supervised and Unsupervised Skill Discovery,” in *International Conference on Learning Representations*, 2020.
- [25] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-End Training of Deep Visuomotor Policies,” *The Journal of Machine Learning Research*, 2016.
- [26] S. Ross, G. Gordon, and D. Bagnell, “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.
- [27] G. Dulac-Arnold, D. Mankowitz, and T. Hester, “Challenges of real-world reinforcement learning,” *arXiv preprint arXiv:1904.12901*, 2019.
- [28] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, “Search on the Replay Buffer: Bridging Planning and RL,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [29] N. Savinov, A. Dosovitskiy, and V. Koltun, “Semi-Parametric Topological Memory for Navigation,” in *International Conference on Learning Representations*, 2018.
- [30] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, “Learning to Explore using Active Neural SLAM,” in *International Conf. on Learning Representations (ICLR)*, 2020.
- [31] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, “Learning Navigation Behaviors End-to-End with AutoRL,” *IEEE Robotics and Automation Letters*, 2019.
- [32] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, “PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5113–5120.
- [33] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, “Scaling Local Control to Large-Scale Topological Navigation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [34] 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.
- [35] ———, “Mobile Robot Navigation using Neural Networks and Nonmetrical Environmental Models,” *IEEE Control Systems Magazine*, 1993.
- [36] A. Francis, A. Faust, H. T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T. W. E. Lee, “Long-Range Indoor Navigation With PRM-RL,” *IEEE Transactions on Robotics*, 2020.
- [37] G. Kahn, P. Abbeel, and S. Levine, “BADGR: An Autonomous Self-Supervised Learning-Based Navigation System,” 2020.
- [38] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Machine Learning*, 1988.
- [39] L. P. Kaelbling, “Learning to achieve goals,” in *IJCAI*. Citeseer, 1993, pp. 1094–1099.
- [40] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.
- [41] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *International Conference on Learning Representations (ICLR)*, 2015.

- [42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” 2017.
- [43] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, “DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [44] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, “Goal-conditioned Imitation Learning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.