IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

# IndoorSim-to-OutdoorReal: Learning to Navigate Outdoors without any Outdoor Experience

Joanne Truong[1,2], April Zitkovich[2], Sonia Chernova[1], Dhruv Batra[1,3], Tingnan Zhang[2], Jie Tan[2], Wenhao Yu[2]

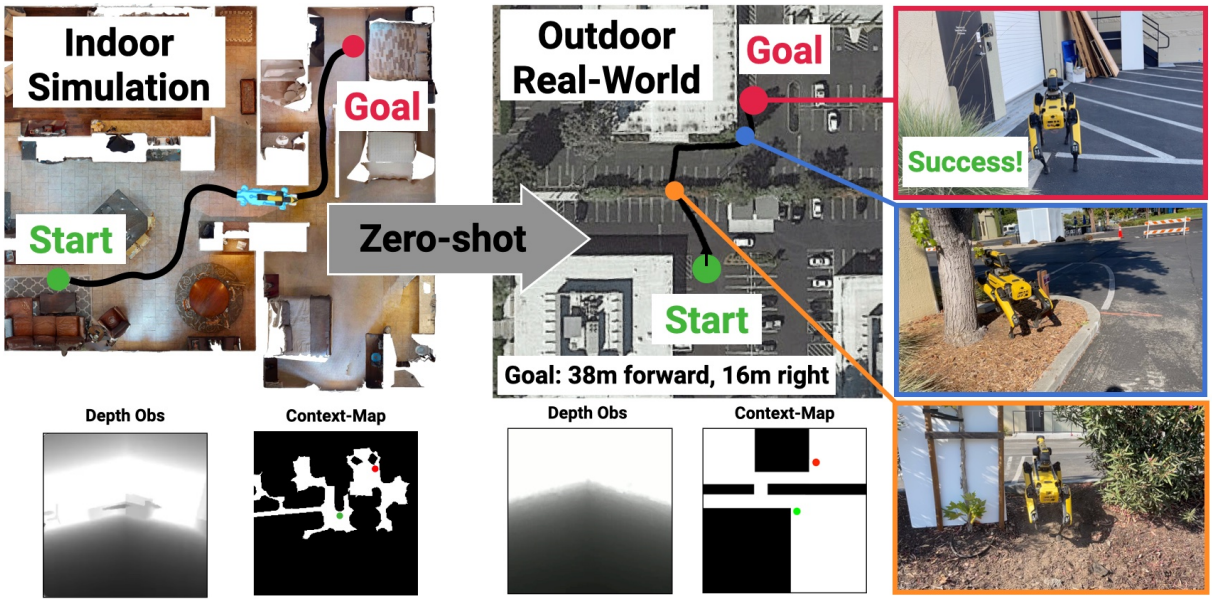


Fig. 1: **Left:** We learn a visual navigation policy using large-scale reinforcement learning in simulated indoor environments. **Right:** We demonstrate zero-shot transfer of this policy for long-range outdoor navigation on the Spot robot. One key ingredient is to combine information from an inaccurate high-level map (titled Context-Map) and accurate-but-limited onboard sensing. The Context-Map is available trivially in indoor-sim and available from satellite imagery or human sketching in outdoor-real experiments. Notice how approximate and incorrect the map is (none of the cars or sidewalks show up on the map), but it does provide a crucial hint to the robot about an opening between the bushes. The robot leverages this hint to navigate up the slope to the opening in the bushes (orange), takes a shortcut between a tree and a wooden pole (blue) that is not marked on the map but visible in egocentric images, to successfully reach the goal position at the entrance of a distant building (red), utilizing the best of an outdated map and up-to-date (but partial) onboard sensing.

***Abstract*— We present IndoorSim-to-OutdoorReal (I2O), an end-to-end learned visual navigation approach, trained solely in simulated short-range indoor environments, and demonstrate zero-shot sim-to-real transfer to the outdoors for long-range navigation on the Spot robot. Our method uses zero real-world experience (indoor or outdoor), and requires the simulator to model no predominantly-outdoor phenomenon (sloped grounds, sidewalks, etc). The key to I2O transfer is in providing the robot with additional context of the environment (*i.e.* a satellite map, a rough sketch of a map by a human, etc.) to guide the robot's navigation in the real-world. The provided context-maps do not need to be accurate or complete– real-world obstacles (*e.g.* trees, bushes, pedestrians, etc.) are not drawn on the map, and openings are not aligned with where they are in the real-world. Crucially, these inaccurate context-maps provide a hint to the robot about a route to take to the goal. We find that our method that leverages Context-Maps is able to successfully navigate over a hundred meters in novel environments, avoiding novel obstacles on its path, to a distant goal without a single collision or human intervention. In comparison, policies without the additional context fail completely. We additionally find that the Context-Map policy is surprisingly robust to noise. In the presence of significantly inaccurate maps in simulation (corrupted with 50% noise, or entirely blank maps), the policy gracefully regresses to the behavior of a policy with no context. Videos are available on our project website.**

## I. Introduction

Recent works in deep reinforcement learning have demonstrated success in training virtual robots to navigate efficiently in simulation before transferring the learned skills to real-world indoor environments [1]–[8], on both wheeled and legged robots. These advances were made possible due to the development of fast, scalable simulators [8]–[18] and the availability of large-scale datasets of photorealistic 3D scans of indoor environments [19]–[21]. Outdoor navigation, in contrast, has been relatively understudied. The complexity of outdoor environments makes it challenging to replicate them accurately in simulations. Consequently, many researchers have resorted to costly data collection over months in the real-world from specific robot embodiments to develop navigation methods. In our work, we aim to overcome this

The Georgia Tech effort was supported in part by NSF, AFRL, DARPA, ONR YIPs, ARO PECASE. JT was supported by an Apple Scholars in AI/ML PhD Fellowship. The views and conclusions are those of the authors and should not be interpreted as representing the U.S. Government, or any sponsor.

[1]JT, SC, and DB are with Georgia Institute of Technology. truong.j@gatech.edu

[2]AZ, TZ, JT, and WY are with Google DeepMind.

[3]DB is with Meta AI.

limitation by leveraging the speed and efficiency of highly optimized indoor simulators, which enables large-scale reinforcement learning. This in turn yields robust navigation policies that can be effectively applied to real-world outdoor environments without the need for environment-specific, high-fidelity simulations.

We present IndoorSim-to-OutdoorReal (I2O), which enables a quadrupedal robot to successfully navigate over a hundred meters in novel outdoor environments, around previously unseen outdoor obstacles (trees, bushes, buildings, pedestrians, etc.) – despite being trained *solely in simulated indoor environments*. The robot has never seen any outdoor environments, and has only been trained using short-range trajectories (on average 8m). We find that the key to enabling I2O is to provide the robot with additional context of its environment (*i.e.* via a satellite image, a rough sketch by a human, etc.), which allows the learned policy to bias its search, obviating the need for costly exhaustive search and backtracking in the real-world. We provide additional context to the robot through a rough human sketch over a satellite image from the environment to guide the robot's navigation. These Context-Maps do not need to be accurate, but serve as a hint in the general directions that the robot should explore. These sketches enable a human user to specify obscure paths that may be difficult for the robot to find otherwise, or paths that are not visible in satellite images (*i.e.* a small opening through bushes shown in Figure 1). It is difficult to apply traditional path planning methods on these inaccurate, incomplete, and outdated Context-Maps. Instead, the robot must learn to use them in conjunction with observations from its onboard cameras to adapt to on-the-ground reality – avoiding obstacles not drawn on the map (bushes, chairs, people).

In our experiments, we find that the robot with the Context-Map is able to navigate over a hundred meters without a single collision or human intervention. In contrast, without the additional context, the robot completely fails at navigating long-ranges outdoors, and is only able to make minor progress towards the goal before getting stuck around obstacles. We conduct a comprehensive quantitative analysis in simulation, and demonstrate that in indoor environments, the additional context can improve success rate by 17%, and improve path efficiency by 22%. Additionally, we find that the Context-Map policy is surprisingly robust to noise in the provided Context-Map. When the maps are inaccurate (corrupted with 50% noise, or an entirely blank map), the performance of the policy gracefully regresses back to the performance of policies trained without any context.

## II. Related Work

**Outdoor Navigation.** Outdoor navigation has a long history in robotics for wheeled and legged robots, and autonomous vehicles. Classical navigation approaches decompose the problem into a sense-plan-act pipeline [22]: a map of the environment is built, the robot is localized within the map, and a planning algorithm is used to generate a path to the goal [23]–[25]. While this classical pipeline has demonstrated successful long-range outdoor navigation [26], [27], these approaches rely on accurate, pre-computed maps of the environment built using expensive LiDAR sensors, and are not adaptable to changes in the environment such as dynamic obstacles. In contrast, our work uses approximate maps to provide a hint to the robot, which are not accurate enough to support planning but contain enough information for a learned policy to improve performance using onboard sensing to adapt to the changes in the environment.

A variety of learning-based methods have been proposed for vision-based outdoor obstacle avoidance and navigation. One prominent method is to leverage offline-learning using large real-world datasets, often collected through teleoperation. Muller et al. [28] utilize an offline dataset and trained a system end-to-end to map short-range observations to steering wheel angles for outdoor obstacle avoidance. Sermanet et al. [29] accomplish long-range, off-road robot navigation by planning at multiple ranges, similar to [30]. Recent works leverage large offline dataset consisting of real-world navigation trajectories collected over months in a wide variety of environments (with the same robot embodiment) using a mix of teleoperation and random walk [31]–[36] for learning navigational affordances. Some works couple this learned module with planning on top of a pre-computed topological map [34], [35]. In contrast, our approach does not require any explicit topological map construction, and the robot can navigate immediately in a new environment without any physical pre-exploration. Shah et al. [36] is the most similar method to ours. They propose to use geographical hints for the task of image goal navigation outdoors. However, their work uses a large offline real-world dataset to reason about traversability. In contrast, our work is learned entirely in simulation and transfers to the real-world zero-shot; no real-world experience or outdoor data is required and our approach is trivially adaptable to new robots because the new robot embodiment can be easily simulated.

An alternative to using real-world datasets is to leverage (outdoor) simulators. Sorokin et al. [37] use CARLA [38], an autonomous driving simulator, to train a quadrupedal robot to navigate on sidewalks by following waypoints generated by public map services (*e.g.* Google Maps). In our work, we remove the assumption of being provided pre-computed waypoints; instead we learn to extract navigational hints directly from the Context-Map and onboard sensing. This enables our robot to navigate using a high-level hint, and take shortcuts through rough terrains and openings when possible. In concurrent work, Sundaresan et al. [39] use hand-drawn sketches for goal specification in manipulation tasks, whereas our work focuses on applying this approach to navigation tasks.

**Indoor Visual Navigation & Sim2real Transfer.** Many works in indoor visual navigation leverage large-scale learning with photorealistic simulators, such as Habitat [9], [10], iGibson [11], [40], or AI2-THOR [13], and techniques for reducing the sim2real gap [41]–[46] to enable zero-shot sim2real transfer on wheeled and legged robots [1]–[8]. We build upon these advances in indoor visual navigation to demonstrate long-range navigation outdoors by leveraging

large-scale learning in short-range simulated indoor environments and outdated maps of the environment.

## III. IndoorSim-to-OutdoorReal Transfer

### A. Experimental Setup

**Task.** In PointGoal Navigation (PointNav) [47], a robot is initialized in a previously unseen environment and needs to navigate to a goal location (*i.e.* "*go to* $\Delta$x, $\Delta$y"). The robot has access to egocentric RGB-D observations, and an egomotion sensor used to calculate the goal location relative to the robot's current pose. The robot operates within constraints of maximum number of steps per episode (10 steps per meter from the goal location) and velocity limits ($\pm 0.5$ $^m/_s$ for linear and $\pm 0.3$ $^{rad}/_s$ for angular velocities). An episode is successful if the robot reaches within 0.425m (half the length of the Spot robot) of the goal location. For evaluation, we report the robot's success rate (SR), and Success weighted by Path Length (SPL) [47], which measures the robot's path efficiency.

**Dataset.** We use the Habitat-Matterport (HM3D) [19] and Gibson [40] 3D datasets, which consist of over 1000 scans of real-world indoor environments (homes, offices, etc.) consisting of realistic clutter (tables, chairs, etc.). We use the training and evaluation navigation episodes from [5], which were generated to result in complex paths (up to 30m).

### B. Context-Guided PointGoal Navigation

We leverage additional context information freely accessible through public map services for the task of long-range PointNav outdoors, which we denote as Context-Guided PointNav (ContextNav). In our experiments, we provide this context in the form of an outdated map (Context-Map).

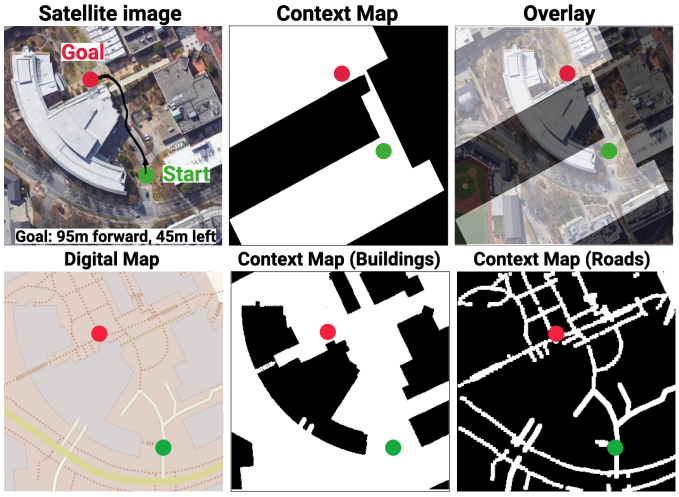


Fig. 2: **Top Row:** Using a satellite image of the area from Google Maps (left), a human operator sketches a rough Context-Map for the robot (middle). Notice how defective the map is (right): large parts or entire buildings are missing, no roads or sidewalks are shown; but crucially, the map contains a hint for an opening to get to the goal. **Bottom Row:** Starting from the digital map (left), we automatically generate Context-Maps by segmenting out buildings (middle), or roads (right).

The Context-Map input does not need to be very accurate, but should serve as a rough guide for general directions that the robot should explore (*i.e.* a satellite map showing roughly where buildings are). Consequently, the robot must use observations from its camera to adapt to novel obstacles or clutter present in the environment but absent on the map, and be willing to take shortcuts available in the world but shown as obstacles on the map. We represent the map as a top-down occupancy map, which can be obtained in the real-world by converting a satellite image to illustrate occupied and freespace through a human sketch, or through an automated process. Using a binary occupancy map provides a few benefits over directly using satellite images. An abstracted binary map allows the maps to be used for both indoor and outdoor navigation, while satellite images are only applicable to outdoor navigation. Additionally, by using human sketches for Context-Maps, the human operator can give hints to the robot about paths that may not be easily visible in the satellite image, or from the robot's initial position (*i.e.* openings in bushes). In our experiments, we use human-sketched Context-Maps (Figure 2 top row, middle), however these maps can easily be generated automatically by postprocessing a digital map using OpenCV thresholding operations (Figure 2 bottom row).

**Policy Architecture.** We train a high-level visual navigation policy entirely in simulation using deep reinforcement learning. We first describe the architecture of a PointNav policy that operates without the additional context (No-Context) from [5], then we outline our extension to the policy architecture to incorporate additional environmental context.

The No-Context policy (Figure 3, left) takes as input an egocentric depth image, a goal vector, and the action at the previous timestep. The output of the policy is the desired center-of-mass linear and angular velocities ($v_x$, $\omega$) for the robot to follow. The goal vector $g_t$ is represented in polar coordinates $[r, \theta]$, representing the distance and heading relative to the robot's current position. Following the recommendation from [48], we first transform the goal vector $[r, \theta]$ to $[r, \cos(\theta), \sin(\theta)]$ to account for the discontinuity at the x-axis in polar coordinates. We then embed the goal vector and previous action $a_{t-1}$ using a fully connected layer, resulting in 32-dimensional outputs. The depth observations are processed using a 3-layer CNN visual encoder resulting in a 512-dimensional output. These depth features $\hat{d}_t$ are fed into a 1-layer Gated Recurrent Unit (GRU) [49] with a 512-dimensional hidden output $h_t$. The output of the GRU is fed into two parallel linear layers with a 2-dimensional output size. The outputs $\mu$ and $\sigma$ parameterize a Gaussian action distribution from which the action is sampled.

Next, we describe how to incorporate additional context information for ContextNav. In simulation, we use freely accessible top-down maps of indoor environments as additional context to aid navigation. The Context-Map is represented as a $2 \times N \times N$ matrix, where $N \times N$ represents the size of the map (we use $N = 100$), and each cell in the map corresponds to $Mcm^2$ in the physical world. We randomize $M$ from [0.05m, 0.5m] during training to improve the robustness of the policy. The first channel of the map

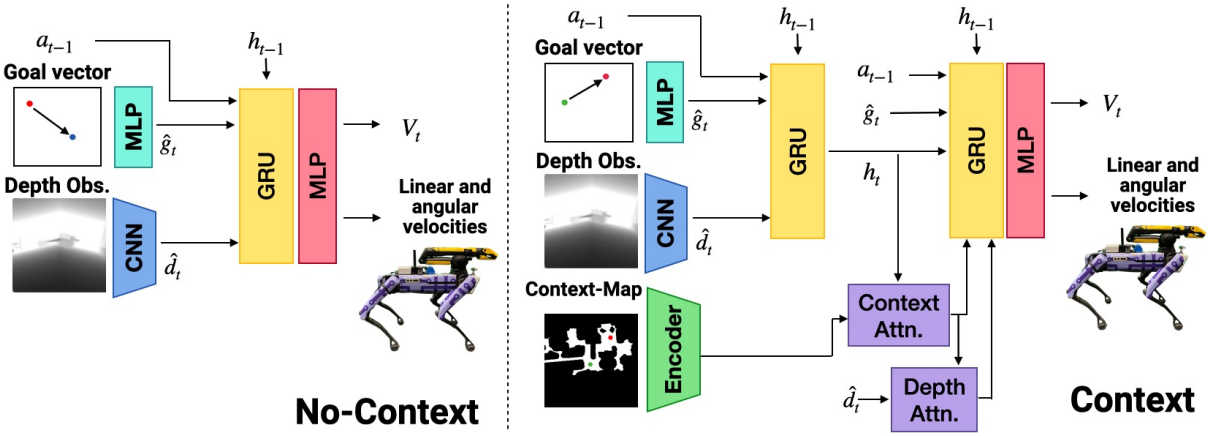


Fig. 3: **Left:** The No-Context PointNav policy architecture consists of a 3-layer CNN to process depth images from the robot's camera, and a MLP to process the goal vector. The policy is a GRU, and outputs linear and angular velocities for the Spot robot to follow and an estimate of the value function. **Right:** The Context-Guided PointNav architecture includes an additional encoder to process the Context-Map. We compute attention between the Context-Map and the depth image, and use a second GRU to process the attended features.

is an egocentric occupancy map, in which the cells denote obstacles (0) or freespace (1). The second channel of the map illustrates the agent's current location in the map and the location of the goal coordinate (1), and 0 otherwise. An example of the second channel is overlaid in green and red in the Context-Map in Figure 3. We process the map context inputs using a ResNet18 visual encoder [50] to obtain a 512-dimensional output. We compute the scaled dot-product attention [51] between the depth and context features. The query is obtained by passing the context features through a linear layer. The key is obtained by passing the context or depth features into a 1D convolution. The value is the context or depth features (depending on the key used). We pass the context and depth attention features, goal features, previous action features, the hidden state from the first GRU and the hidden state from the previous timestep into a second GRU.

**Training details.** We train all our policies for 500M steps using DD-PPO [48], a distributed reinforcement learning method, in the Habitat simulator [9], [10]. The reward function is derived from [5], and is defined as:

$$r_t = r_{success} + r_{geo} + r_{slack} + r_{backward} + r_{coll} \quad (1)$$

$r_{success}$ is a terminal reward for successful episodes. $r_{geo}$ is a dense reward for following the shortest path to the goal. $r_{slack}$ is a slack penalty to incentivize the robot to reach the goal quickly. $r_{backward}$ is a penalty for backward velocities. $r_{coll}$ is a collision penalty. We set $r_{success}$ to 10.0, $r_{slack}$ to -0.002, $r_{backward}$ to -0.3, and $r_{coll}$ to -0.003.

### C. Techniques to Aid IndoorSim-to-OutdoorReal Transfer.

Empirically, we found that several key techniques were needed to enable IndoorSim-to-OutdoorReal transfer. We use these techniques for all policies tested in the real-world.

**Indoor-to-Outdoor Transfer.** Two of the main differences between indoor and outdoor navigation are 1) navigation length (short-range vs. long-range), and 2) terrain type (flat vs. rocky/sloped). First, we found that traditional PointNav policies were highly sensitive to the goal vector. Since the policies were only trained in simulation, and typically see trajectories ∼8m away, the policies failed to generalize to longer-range goals. We normalize the goal vector by using the log of the goal distance, which enabled longer-range navigation. Next, we found that naively trained PointNav policies had difficulty navigating up slopes. Since slopes are infrequent in indoor environments, slopes outdoors appear as a large obstacle in the robot's depth camera, and thus the robot avoids walking up the slope. To enable the robot to walk up and down slopes in the real-world, we artificially add slopes to the robot's observation by randomizing the pitch of the camera during training by $\pm 30°$.

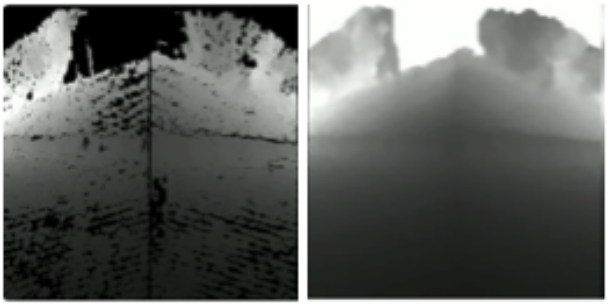

Fig. 4: **Left:** Raw depth images from Spot's cameras. **Right:** Filtered depth images using [52] to match images from simulation.

**Sim-to-Real Transfer.** In simulation, instead of modeling the robot's low-level locomotion control, we use kinematic control as an approximation for the robot's movement. In kinematic control, the robot is moved to its next state via Euler integration at 2Hz without running full rigid-body physics. If the robot were to collide at the next state, we simply keep the robot in place. Kinematic control was shown in [5] to lead to better sim-to-real transfer through faster simulation, as compared to dynamic control. Next, we filter the depth images from Spot's depth cameras in the real-world to better match observations from simulation. The raw depth images from Spot are show in Figure 4 left. We use depth completion from [52] to fill in holes and smooth the image. Additionally, we set the pixels further away from the max depth range (3.5m) to white to match the depth images from simulation. The filtered depth image is shown in Figure 4


Fig. 5: We test our policies in 3 novel environments in the real-world. The routes contain many obstacles including bushes and buildings, that the robot has never seen during training. The Context-Map policies (black) are able to navigate hundreds of meters to reach the goal 100% of the time. In comparison, the No-Context policies (blue) beeline towards the goal and get stuck on obstacles, resulting in failures.

right. Lastly, to improve the robustness of our policies to missing pixels in real-world depth images, we add Random Erasing noise [53] to our depth observations during training.

### D. Real-world Setup

We use the Boston Dynamics (BD) Spot robot in simulation and the real-world. Our navigation policy outputs high-level velocity commands, and we rely on BD's low-level controller for movement. We use Spot's two front-facing Intel Realsense D430 depth cameras for visual inputs to our policy. We disable the BD obstacle avoidance in order to isolate the performance of our policy from confounding factors; however, a human operator was vigilant to terminate the episode (and report it as a failure) if a collision were imminent. The BD API includes two modes for high-level navigation, but both methods have limitations and cannot be used directly for long-range outdoor navigation.

## IV. Results

In this section, we first study zero-shot sim2real indoor2outdoor navigation. Next, we conduct extensive experiments in simulation to systematically quantify the value of using the additional context input, and the robustness of the policy to noise in the Context-Maps.

TABLE I: We test the No-Context and Context-Map policies on 3 long-range outdoor routes (Figure 5).

| Route # | Goal | Method | SR ↑ | Distance Travelled (m) ↑ |
|---|---|---|---|---|
| 1 | 38m Forward, 16m Right | No-Context | 0.0 | 16.6$\pm$0.1 |
| | | Context-Map | **100.0** | **63.4**$\pm$2.5 |
| 2 | 90m Forward, 30m Left | No-Context | 0.0 | 9.7$\pm$3.4 |
| | | Context-Map | **100.0** | **112.2**$\pm$1.8 |
| 3 | 95m Forward, 45m Left | No-Context | 0.0 | 5.1$\pm$0.3 |
| | | Context-Map | **100.0** | **129.8**$\pm$2.8 |

### A. Zero-shot IndoorSim-to-OutdoorReal Navigation

We task the robot with navigating to 3 long-range goals outdoors (shown in Figure 5), with many real-world obstacles present (bushes, buildings, cars, tables, pedestrians, etc.). We report the average success rate (SR) and distance travelled across 3 runs for each method in Table I.

**No-Context Real-world Outdoor Navigation.** First, we test to see if policies with No-Context can directly transfer to long-range navigation outdoors. The outdoor routes are complex and there does not exist a straight-line path to the goal; the robot is required to navigate around large obstacles to reach the goal successfully. We find that the No-Context policy immediately makes a beeline towards the goal using the goal vector for guidance. In indoor environments, there are more obstacles around the robot, such as walls, that guide the robot to avoid making a beeline to the goal. In the outdoors however, the robot makes a beeline presumably led by the depth sensors that indicate plenty of free-space around the robot. This leads the robot to wander into obstacles such as bushes, resulting in unsuccessful episodes. The No-Context policy completely fails to navigate in all three routes, each only making minor progress to the goal before an operator had to intervene to prevent a collision (Table I). The trajectories taken by the No-Context policy is shown in blue in Figure 5. In the first route, the shortest path requires passing through a small opening to the left of the robot between two bushes that is not visible to the robot from the start. Since the goal vector indicates that the goal is to the right, the robot walks to the right and gets stuck searching by the bushes, resulting in an episode failure.

**Context-Guided Real-world Outdoor Navigation.** Next, we test our Context-Map policy outdoors. We provide the robot with rough map sketches to guide its navigation for each route (Context-Map in Figure 5). Notice how rough and incomplete the provided maps are– obstacles such as cars, trees, or chairs are not shown on the map, and only rough hints for an opening to the goal is depicted. We find that the Context-Map policy is able to leverage the Context-Maps to bias its search during outdoor navigation, and successfully reach the distant goal location 100% of the time (Table I), without a single collision or human intervention. Shah et al. [36] report similar results using

42 hours of real-world robot navigation data. In contrast, our approach uses zero real-world experience. With our approach, the robot was able to navigate around dynamic obstacles such as pedestrians and real-world clutter despite these obstacles not being drawn directly on the Context-Map. Our approach is also not limited to 2D navigation; the robot navigates up slopes (Route 1), and can navigate in various terrains including dirt slopes and grass.

We find that our policies are able to maintain a balance between the Context-Maps, and what it sees in its visual inputs. The robot leverages depth images to avoid clutter or dynamic obstacles, and Context-Maps are used for high-level guidance. While Context-Maps provide a means for the operator to specify a preferred route for the robot to take, the robot may take shortcuts along the way that are not present in the map when its vision senses freespace. However, we observe that the robot is unable to find a drastically 'better' solution than what is hinted on the map (*i.e.*, a better path to the right, when the map indicates free space only on the left), due to the robot's limited vision (recall that the depth cameras have a range of only ∼3.5m).

**Traditional Planning Baseline.** We compare our approach against a planning baseline. We first experiment in simulation using oracle RRT* [3], [54] mapping + online re-planning. Specifically, we start with a map consisting entirely of freespace, and update this map with the ground-truth top-down map based on the robot's current position and field of view, up to the robot's maximum depth (3.5m). This map is fed into RRT* to generate waypoints to the goal. We use the waypoint from RRT* to determine the next best action for the robot to take. We find that using RRT* to re-plan at each step is prohibitively slow, as it is important to let RRT* converge on a plan at each step before re-planning ($>$ 22 seconds per step). In contrast, our approach uses a neural network that takes on average 6 *ms* at each step. As the size of the scene increases (and goal distance increases), the time to run RRT* to convergence will increase as well, while our policy remains fast yet robust. Since online planning with RRT* is prohibitively slow, we experiment with using RRT* for planning offline. We run RRT* on the *exact same* outdated Context-Maps used for our Context-Map policy to generate a list of waypoints to the goal. These waypoints are then used with a policy trained in simulation to follow waypoints along the shortest route to the goal. We find that the policy is able to successfully navigate to each successive waypoint. However, the robot ended up missing the actual opening in bushes because the waypoints were generated from an outdated map (Figure 6). This demonstrates the difficulty in using offline planning approaches on inaccurate maps. In the real-world, creating a perfect, and always-up-to-date map is not realistic.

### *B. Simulation Results*

**Indoor Navigation.** We evaluate using 1200 episodes of indoor environments from the HM3D and Gibson scenes [5], and report the average across 3 seeds in Table II. These navigation episodes are complex, and require navigating through multiple cluttered indoor rooms. We find that the

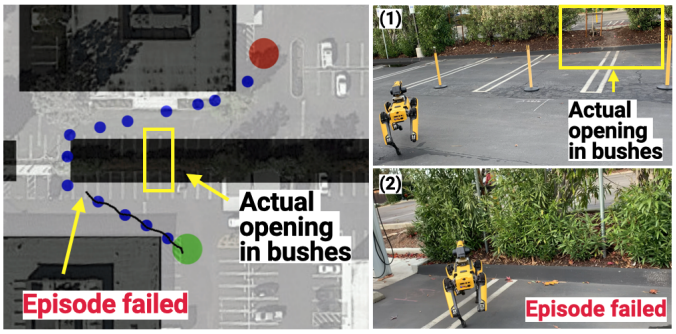


Fig. 6: **Left:** We run RRT* on an outdated Context-Map to find waypoints (blue) to the goal. We pass the waypoints to a policy to follow (black). **Right:** The waypoints lead the robot past the opening in the bushes (1), leading to a failure (2).

Context-Map policies achieve high success rate and SPL (95.4 SR and 81.0 SPL; Table II, row 2), demonstrating that the policy is able to utilize its given Context-Map to navigate to the goal. In contrast, the No-Context policies achieve a lower success rate of 78.7% (-16.7% SR), often failing by exceeding the number of steps allowed within an episode, caused by getting stuck on obstacles or taking the wrong turn to the goal. In contrast, our Context-Map policy is able to leverage the information in the provided Context-Map to guide its navigation to take efficient paths to the goal, resulting in a +24.6% SPL between the two policies

TABLE II: We evaluate the No-Context and Context-Map policies in simulation of indoor environments and report the average success rate (SR) and success weighted by path length (SPL).

| Method | SR ↑ | SPL ↑ |
|---|---|---|
| No-Context | 78.7±8.2 | 56.4±4.4 |
| Context-Map | 95.4±0.2 | 81.0±2.4 |

**Indoor Navigation using Outdated Maps.** We test the robustness of our policy by adding varying degrees of noise to the Context-Map, shown in Figure 7. We use: (1) Shift noise, which randomly shifts the map in any direction to simulate localization errors that may occur. (2) Rotation noise, which randomly rotates the map in any direction to simulate heading errors that may occur. (3) Cutout noise, which randomly adds patches of free or obstacle space to the map. This simulates giving the robot an outdated map (*i.e.* a map with additional or missing obstacles in the environment).

We find that our approach, trained entirely using perfect maps, is surprisingly robust to the noisy maps. At 50% cutout noise, the original top-down map is barely visible (Figure 7), and planning algorithms would be unable to find any feasible path to the goal. In contrast, our policy is still able to navigate to the goal successfully 81.4% of the time (Table III, row 8). When the Context-Map policy is given a map that is completely freespace (100% noise, Table III row 2) the policy behavior regresses to the No-Context policy (79.8 vs. 78.7 SR, and 59.8 vs. 56.4 SPL). Thus, our approach exhibits the best of both worlds – utilizing the information in the map when it is available but never underperforming a map-free approach. We attribute the robustness of our policy

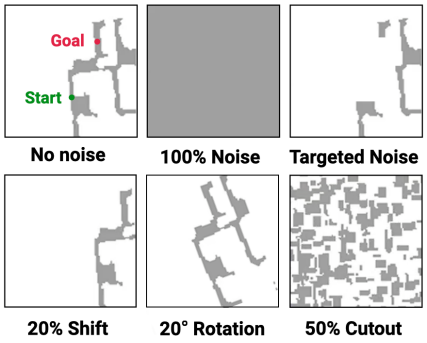


Fig. 7: We use shift, rotation, cutout, and targeted noise to degrade the top-down map obtained in simulation.

to the scale and diversity during training. We additionally train policies with noisy maps (Table III rows 9-14), and find that policies trained with noise are even more robust to noisy maps. This reinforces the idea that training diversity can improve generalization of the navigation policy.

TABLE III: We evaluate with varying degrees of noise to the map.

| # | Train Noise | Eval Noise | SR ↑ | SPL ↑ |
|---|---|---|---|---|
| 1 | - | - | 95.4 ±0.2 | 81.0 ±2.4 |
| 2 | - | 100% Noise | 79.8 ±1.2 | 59.8 ±0.9 |
| 3 | - | 5% Shift | 93.0 ±0.9 | 75.1 ±2.4 |
| 4 | - | 10% Shift | 84.0 ±0.5 | 62.5 ±2.3 |
| 5 | - | 20% Shift | 73.2 ±1.6 | 51.6 ±1.1 |
| 6 | - | 5° Rotation | 87.9±1.5 | 71.9±2.2 |
| 7 | - | 10° Rotation | 84.8±3.2 | 68.5±3.6 |
| 8 | - | 20° Rotation | 80.7±2.8 | 62.1±3.4 |
| 9 | - | 10% Cutout | 91.5 ±0.2 | 72.3 ±2.2 |
| 10 | - | 25% Cutout | 87.6 ±0.8 | 67.4 ±2.3 |
| 11 | - | 50% Cutout | 81.4 ±2.2 | 60.0 ±1.0 |
| 12 | 20% Shift | 5% Shift | 94.0±1.7 | 73.7±5.6 |
| 13 | 20% Shift | 10% Shift | 93.8±1.6 | 73.9±6.0 |
| 14 | 20% Shift | 20% Shift | 93.3±1.6 | 73.6±6.2 |
| 15 | 50% Cutout | 10% Cutout | 91.6±1.6 | 72.7±3.1 |
| 16 | 50% Cutout | 25% Cutout | 92.5±1.1 | 73.1±3.5 |
| 17 | 50% Cutout | 50% Cutout | 92.2±1.1 | 72.6±3.1 |

We further investigate what parts of the Context-Map are most crucial for effective navigation. Instead of randomly adding either shift or cutout noise, we apply a 'targeted-noise map' as input to the Context-Map policy by masking out the pixels in the Context-Map between the robot's current position and the goal position (marking the pixels as obstacles). We believe that that the value of the Context-Map is to provide the policy with high-level directions on where to start exploring. Thus, by adding this targeted noise, we remove any hints or high-level directions to the goal. We find that with this targeted noise, the overall performance drops to 76.3±0.9 SR and 53.8 ±4.2 SPL (-5.1% SR and -6.2% SPL compared to 50% cutout noise). This demonstrates that when the Context-Map no longer contains a high-level 'hint' to the path to the goal, the Context-Map is no longer useful to the policy, and instead the robot must use information from its visual input for navigation. This performs similarly to the No-Context policy (78.7 SR, 56.4 SPL). This reinforces the benefit of our approach– the policy is able to utilize the information in the map when it's available, but can fall back to navigating using vision (as a map-free approach would do) if the map no longer provides a hint to the goal.

## V. Discussion & Limitations

Our results provide compelling evidence for rejecting the (admittedly reasonable) hypothesis that a new simulator must be designed for every new scenario we wish to study. This is especially important for environments that are challenging to design in simulation, such as the outdoors. Traditional approaches for 'virtualizing' a real-world environment to import into a simulator leverage infrared (IR) projection based 3D scanners. However, in the outdoors, IR light interference from the sun prohibits scanning these environments accurately. Instead, our approach shows that simulators can be used for zero-shot real-world transfer for long-range navigation outdoors without having to design and model the deployment scenario apriori. Our work uses zero real-world experience (indoor or outdoor) and requires the simulator to model no predominantly-outdoor phenomenon (terrain, ditches, sidewalks, cars, etc). The power in the approach comes from a combination of a robust locomotion system on Spot (which itself uses a well-designed classical robotics stack), low sim-vs-real gap in depth and map sensors, and the power of large-scale learning.

Our approach suffers from a few limitations. First, there may be instances in which both the depth observation and Context-Map may suggest freespace in front of the robot despite an obstacle being present (narrow railings, glass walls, etc.), which would cause the robot to fail. To avoid such failures, it is important to incorporate other sensing modalities such as proprioception input [55], or to endow the robot with additional actions (look up or down, etc.) to increase the robot's sensing. Second, our approach relies on the BD API for localization, which may drift over time as the robot navigates longer distances. In the future, we would like to remove the assumption of accurate localization sensors. Lastly, while our approach demonstrates that large-scale learning with simulated data alone can enable outdoor navigation, prior works have also demonstrated impressive results using real-world data. In the future we would like to combine learning with both real-world and simulated data. Training a policy from scratch in the real-world is intractable and may cause damage to the robot over time, but it may be beneficial to finetune a pre-trained policy in the real-world. An alternative is to leverage offline trajectories from the real-world to augment the simulator [45], [56].

## VI. Conclusion

In this work, we present a framework for IndoorSim-to-OutdoorReal transfer for navigation. Our navigation policy is trained solely in simulated short-range trajectories from indoor datasets, yet is able to navigate long ranges outdoors in the real-world. The navigation policy is robust to novel environments and obstacles such as sloped grounds, bushes, and trees never seen during training. Keys to the system's success are a set of sim-to-real techniques that

enable the policy to handle real outdoor environments, as well additional context of the environment, which guides the robot's navigation. This work opens up using simulation for navigation research in outdoor environments.

## REFERENCES


[1] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, *et al.*, "Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?" in *RA-L*, 2020.

[2] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, *et al.*, "Learning Navigation Skills for Legged Robots with Learned Robot Embeddings," in *IROS*, 2020.

[3] N. Yokoyama, S. Ha, and D. Batra, "Success Weighted by Completion Time: A Dynamics-Aware Evaluation Criteria for Embodied Navigation," in *IROS*, 2021.

[4] R. Partsey, E. Wijmans, N. Yokoyama, O. Dobosevych, D. Batra, *et al.*, "Is Mapping Necessary for Realistic PointGoal Navigation?" in *CVPR*, 2022.

[5] J. Truong, M. Rudolph, N. Yokoyama, S. Chernova, D. Batra, *et al.*, "Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation," in *CoRL*, 2022.

[6] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning To Explore Using Active Neural SLAM," in *ICLR*, 2020.

[7] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining Optimal Control and Learning for Visual Navigation in Novel Environments," in *CoRL*, 2020.

[8] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, *et al.*, "Robothor: An Open Simulation-to-Real Embodied AI Platform," in *CVPR*, 2020.

[9] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, *et al.*, "Habitat: A Platform for Embodied AI Research," *ICCV*, 2019.

[10] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, *et al.*, "Habitat 2.0: Training Home Assistants to Rearrange their Habitat," in *NeurIPS*, 2021.

[11] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, *et al.*, "iGibson 1.0: A Simulation Environment for Interactive Tasks in Large Realistic Scenes," in *IROS*, 2021.

[12] Nvidia, "Isaac Sim," https://developer.nvidia.com/isaac-sim, 2020.

[13] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, *et al.*, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv*, 2017.

[14] C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, *et al.*, "ThreeDWorld: A Platform for Interactive Multi-modal Physical Simulation," *NeurIPS Dataset*, 2021.

[15] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, *et al.*, "SAPIEN: A SimulAted Part-based Interactive ENvironment," in *CVPR*, 2020.

[16] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A Physics Engine for Model-based Control," in *IROS*, 2012.

[17] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, *et al.*, "Brax–A Differentiable Physics Engine for Large Scale Rigid Body Simulation," *NeurIPS Dataset*, 2021.

[18] E. Coumans and Y. Bai, "Pybullet, A Python Module For Physics Simulation For Games, Robotics, And Machine Learning," 2016.

[19] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, *et al.*, "Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI," *NeurIPS Datasets and Benchmarks Track*, 2021.

[20] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, *et al.*, "Matterport3D: Learning from RGB-D Data in Indoor Environments," *International Conference on 3D Vision (3DV)*, 2017.

[21] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University, Princeton University, Toyota Technological Institute at Chicago, Tech. Rep., 2015.

[22] R. R. Murphy, *Introduction to AI robotics*. MIT press, 2019.

[23] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping," *IEEE robotics & automation magazine*, 2006.

[24] S. Thrun, "Probabilistic Robotics," *ACM*, 2002.

[25] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual Simultaneous Localization and Mapping: a Survey," *Artificial intelligence review*, 2015.

[26] Y. Morales, A. Carballo, E. Takeuchi, A. Aburadani, and T. Tsubouchi, "Autonomous Robot Navigation in Outdoor Cluttered Pedestrian Walkways," *Journal of Field Robotics*, 2009.

[27] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "Autonomous Robot Navigation in Highly Populated Pedestrian Zones," *Journal of Field Robotics*, 2015.

[28] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun, "Off-Road Obstacle Avoidance Through End-to-End Learning," *Advances in neural information processing systems*, 2005.

[29] P. Sermanet, R. Hadsell, M. Scoffier, M. Grimes, J. Ben, *et al.*, "A Multirange Architecture for Collision-Free Off-Road Robot Navigation," *Journal of Field Robotics*, 2009.

[30] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, *et al.*, "Learning Long-Range Vision for Autonomous Off-Road Driving," *Journal of Field Robotics*, 2009.

[31] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An Autonomous Self-Supervised Learning-Based Navigation System," *IEEE RA-L*, 2021.

[32] G. Kahn, P. Abbeel, and S. Levine, "Land: Learning to Navigate from Disengagements," *IEEE Robotics and Automation Letters*, 2021.

[33] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, "Rapid Exploration for Open-World Navigation with Latent Goal Models," in *CoRL*, 2021.

[34] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "VING: Learning Open-World Navigation with Visual Goals," in *ICRA*, 2021.

[35] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A General Navigation Model to Drive Any Robot," in *ICRA*, 2023.

[36] D. Shah and S. Levine, "ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints," in *RSS*, 2022.

[37] M. Sorokin, J. Tan, K. Liu, and S. Ha, "Learning to Navigate Sidewalks in Outdoor Environments," *RA-L*, 2022.

[38] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[39] P. Sundaresan, Q. Vuong, J. Gu, P. Xu, T. Xiao, *et al.*, "RT-Sketch: Goal-Conditioned Imitation Learning from Hand-Drawn Sketches," 2023.

[40] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, *et al.*, "Gibson Env: Real-World Perception for Embodied Agents," in *CVPR*, 2018.

[41] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, *et al.*, "Sim-to-Real: Learning Agile Locomotion for Quadruped Robots," in *RSS*, 2018.

[42] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, *et al.*, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *IROS*. IEEE, 2017.

[43] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real Transfer of Robotic Control with Dynamics Randomization," in *ICRA*, 2018.

[44] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg, "Dynamics Randomization Revisited: A Case Study for Quadrupedal Locomotion," in *ICRA*. IEEE.

[45] J. Truong, S. Chernova, and D. Batra, "Bi-directional Domain Adaptation for Sim2Real Transfer of Embodied Navigation Agents," in *RA-L*, 2021.

[46] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, *et al.*, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," in *ICRA*, 2018.

[47] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, *et al.*, "On Evaluation of Embodied Navigation Agents," *arXiv preprint arXiv:1807.06757*, 2018.

[48] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, *et al.*, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," *ICLR*, 2020.

[49] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, *et al.*, "Attention is All You Need," *NeurIPS*, 2017.

[52] J. Ku, A. Harakeh, and S. L. Waslander, "In Defense of Classical Image Processing: Fast Depth Completion on the CPU," in *CRV*, 2018.

[53] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," in *AAAI*, 2020.

[54] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, 2011.

[55] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, *et al.*, "Coupling Vision and Proprioception for Navigation of Legged Robots," *CVPR*, 2022.

[56] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer, "Sim-to-Real Transfer with Neural-Augmented Robot Simulation," in *CoRL*, 2018.