Sim-to-Real Transfer for Vision-and-Language Navigation

Peter Anderson[1][*] **Ayush Shrivastava**[1] **Joanne Truong**[1] **Arjun Majumdar**[1]
Devi Parikh[1,2] **Dhruv Batra**[1,2] **Stefan Lee**[3]

[1]Georgia Institute of Technology [2]Facebook AI Research [3]Oregon State University

Abstract: We study the challenging problem of releasing a robot in a previously unseen environment, and having it follow unconstrained natural language navigation instructions. Recent work on the task of Vision-and-Language Navigation (VLN) has achieved significant progress in simulation. To assess the implications of this work for robotics, we transfer a VLN agent trained in simulation to a physical robot. To bridge the gap between the high-level discrete action space learned by the VLN agent, and the robot's low-level continuous action space, we propose a subgoal model to identify nearby waypoints, and use domain randomization to mitigate visual domain differences. For accurate sim and real comparisons in parallel environments, we annotate a $325m^2$ office space with 1.3km of navigation instructions, and create a digitized replica in simulation. We find that sim-to-real transfer to an environment not seen in training is successful if an occupancy map and navigation graph can be collected and annotated in advance (success rate of 46.8% vs. 55.9% in sim), but much more challenging in the hardest setting with no prior mapping at all (success rate of 22.5%).

1 Introduction

We study the challenging problem of releasing a robot in a previously unseen environment, and having it follow unconstrained natural language navigation instructions. Most previous evaluations of instruction-following robots either focus on smaller table-top environments [1–3], or are evaluated in simulation [4–12]. However, performing only component-level evaluation (e.g., of the instruction parser) or evaluating only in simulation neglects real-world sensing, actuation and localization errors and the challenges of integrating complex components, which may give a misleading impression of progress. Therefore, leveraging the cumulative advances of previous authors studying Vision-and-Language Navigation (VLN) in simulation [13–20], we transfer a VLN agent trained in simulation on the R2R dataset [13] to a physical robot and complete one of the first full system evaluations of a robot following unconstrained English language directions in an unseen building.

As illustrated in Figure 1, VLN [13] is a formulation of the instruction-following problem that requires an agent to interpret a natural-language instruction and then execute a sequence of actions to navigate efficiently from the starting point to the goal in a previously unseen environment. In existing working studying VLN in simulation, the agent's action space is typically defined in terms of edge traversals in a navigation graph, where nodes are represented with 360° panoramic images (on average 2.1m apart) and edges indicate navigable paths between these panoramas. In effect, high visual fidelity comes at the cost of low control fidelity. Given this limitation, a major challenge with sim-to-real transfer of a VLN agent is bridging the gap between the high-level discrete action space learned by the agent and the low-level continuous physical world in which the robot operates.

To address this challenge, we propose a subgoal model that, conditioning on both 360° RGB images and laser scans, identifies a set of navigable nearby waypoints that can be evaluated by the VLN agent as high-level action candidates. To minimize domain differences between sim and real, the subgoal model is trained on the same navigation graphs (from the R2R / Matterport3D [21] dataset) as the VLN agent. To support navigation to the waypoint selected by the VLN agent, we assemble a classical navigation stack based on Robot Operating System (ROS) [22], incorporating a standard
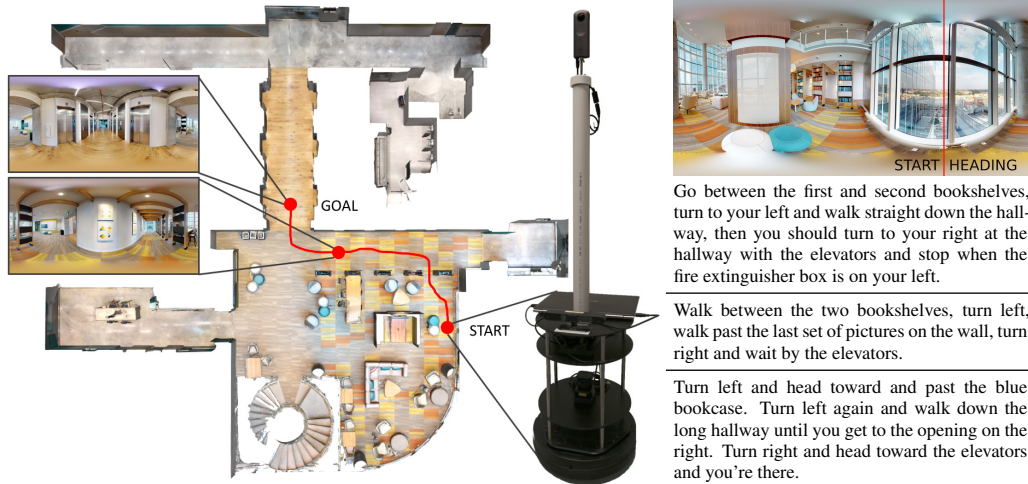
---
[*]Now at Google.

Go between the first and second bookshelves, turn to your left and walk straight down the hallway, then you should turn to your right at the hallway with the elevators and stop when the fire extinguisher box is on your left.

Walk between the two bookshelves, turn left, walk past the last set of pictures on the wall, turn right and wait by the elevators.

Turn left and head toward and past the blue bookcase. Turn left again and walk down the long hallway until you get to the opening on the right. Turn right and head toward the elevators and you're there.

Figure 1: We transfer a VLN agent trained in simulation to a physical robot (center) placed in a 325m^2 office environment not seen in training (left). Our experiments compare instruction-following performance in parallel simulator and physical environments over 111 unconstrained natural language navigation instructions (e.g., the 3 instructions on right corresponding to the red target trajectory).

Simultaneous Localization and Mapping (SLAM) implementation [23] along with obstacle avoidance and path-planning routines. For sim-to-real experiments we use a TurtleBot2 mobile robot equipped with a 2D laser scanner and a 360° RGB camera. To mitigate the impact of visual differences between sim and real, we train the VLN agent and the subgoal model using domain randomization [24].

Evaluations are conducted in a 325m^2 physical office environment we refer to as Coda. Since our focus is evaluating whether progress on VLN in simulation can be parlayed into progress in robotics, we digitize and annotate the Coda environment to provide a reliable comparison of sim and real performance in parallel unseen test environments. Specifically, a Matterport camera is used to scan and reconstruct Coda and the resulting assets are imported into the Matterport3D simulator. We then construct a dataset of English language instruction-trajectory pairs in Coda using Amazon Mechanical Turk (AMT) and following the R2R data collection protocols. In total, we collect 111 navigation instructions representing 1,334m of language-guided trajectories.

Experimentally, we complete two full physical evaluations in Coda, testing two settings: 'with map' in which an occupancy map and navigation graph are collected and annotated in advance, and 'no map' in which the robot performs SLAM from scratch each time a new instruction is received. In all evaluations the language inputs and the visual appearance of the environment are previously unseen. We show that sim-to-real transfer is reasonably successful in the 'with map' setting (success rate of 46.8% vs. 55.9% in simulation, with reductions in success rate attributed 3.9% to remaining visual domain differences and 5.2% to viewpoint differences). However, in the hardest setting with no prior mapping, sim-to-real transfer is much less reliable (success rate of 22.5%) due to subgoal prediction errors which fail to fully abstract the differences in the agent's action space between sim and real.

Contributions. In summary, we achieve the first sim-to-real transfer of a VLN agent trained in simulation on the R2R dataset to a robotic platform. Our main contributions include:

- A new annotated VLN simulator environment corresponding to an accessible physical environment,
- A sim-to-real framework interfacing a trained VLN agent with standard ROS components,
- A subgoal prediction model to bridge the gap between the discrete action space learned by the VLN agent and the continuous world, and
- An empirical study quantifying the sim-to-real gap in terms of errors due to visual domain differences, viewpoint differences and subgoal prediction errors / action space differences.

We provide code[2] for digitizing and annotating new VLN environments, plus our sim-to-real framework and subgoal model for deploying VLN agents to robots using ROS.

---

[2]https://github.com/batra-mlp-lab/vln-sim2real

2 Related Work

Instruction-Following Robots Although natural language command of robots in unstructured environments could be considered a grand challenge of robotics, there is surprisingly little previous work in our setting of interest – in which a physical robot is released in a building it hasn't seen in training and evaluated on it's ability to execute unconstrained natural language navigation instructions. The most similar settings to ours are (noting major differences from ours in parentheses): the voice-commandable wheelchair of Hemachandra et al. [25] (relies on artificial landmarks), the quadcopters of Huang et al. [26] and Blukis et al. [27] (evaluated in the training environment), and the voice-commanded robot teammate of Oh et al. [28] (outdoors). Most other evaluations of language-guided robots have been limited to only a handful of instruction commands [29, 30] or a handful of object types [31, 32], or they focus on manipulation rather than navigation [33, 34].

Vision-and-Language Navigation (VLN) In the VLN task [13], an agent is placed in a photo-realistic simulation of an indoor environment and given a natural language navigation instruction describing the path to the goal. To reach it, agents must learn to ground language instructions to both visual observations and actions. In the standard setting, the test environments are unseen during training and no prior exploration is permitted. Despite the task's difficulty, recent work has seen significant improvements [14–20], including the use of pragmatic speaker models for trajectory re-ranking and data augmentation [18, 19], as well as progress estimation [16] and backtracking [17, 20].

Pano-Simulators and Datasets A growing number of simulation environments, tasks and datasets have been proposed based on situated panoramic images. For example, building on R2R/Matterport3D [13, 21], annotations for vision-and-dialog navigation [35], asking for help [36], remote embodied referring expressions [37], and multilingual VLN [38, 39] have been released. In the outdoor setting, several panoramic image datasets have been proposed including StreetLearn [40, 41] and SEVN [42], giving rise to language navigation datasets such as TouchDown [43], Talk2Nav [44] and RUN [45]. With the increasing interest in training embodied agents in panoramic image environments, there is an urgent need to investigate the transfer of these agents to real physical platforms.

3 Sim-to-Real Experimental Setting

Coda Test Environment For evaluation of the VLN robot we select Coda as the unseen test environment. Coda is a 325m^2 collaborative shared space in a commercial office building. As a shared space, Coda is devoid of personal items such as papers, posters, photos, bags and computing devices. While this eliminates some interesting visual clutter, it also helps minimize the drift between the static simulator and the real physical environment over time. Visual diversity is enhanced by the variety of rooms included, such as an elevator lobby, several long corridors, a lounge area and various break-out spaces (refer floorplan in Figure 1), while floor-to-ceiling glass walls and windows provide reflections and changing lighting that makes the space particularly challenging for robotic vision.

Simulator Construction To accurately establish the sim-to-real gap in the unseen test environment, we construct a parallel simulator environment by reconstructing Coda with a Matterport3D Pro 2 camera and the Matterport3D web services. We download the resulting pointcloud and textured mesh, plus the equirectangular panoramic image and pose at each of the camera 65 viewpoints, and a 'visibility graph' indicating which pairs of camera viewpoints are mutually visible. After excluding 6 viewpoints located on stairs or above furniture (which are unreachable by our robot), following Anderson et al. [13] we construct a navigation graph from the visibility graph by excluding edges with length greater than 5m. The resulting navigation graph, panoramic images and viewpoint poses are then imported into the Matterport3D simulator to create the Coda simulator environment.

Navigation Instructions To collect navigation instructions for Coda, we follow Anderson et al. [13] by sampling trajectories that are the shortest path between two points and then asking annotators to describe these paths using an immersive 3D web interface. To validate instruction quality, we then ask a different annotator to follow each instruction using a similar interface. In total we sample 37 trajectories and collect four English language navigation instructions for each trajectory using Amazon Mechanical Turk (AMT). After discarding the instruction with the highest follower navigation error for each trajectory, the final Coda dataset contains 111 instructions, representing a total of 1,334m of language-guided navigation with an average human follower success rate of 93%

(vs. 86% for the R2R test set). As indicated by the example in Figure 1, the resulting trajectories and instructions are similar in length and style to R2R, with an average of 25 words per instruction (vs. 29 for R2R). The vocabulary size of the Coda instructions is in the 44th percentile of R2R environment vocabularies (based on repeatedly randomly sampling 37 paths / 111 instructions per R2R environment, and skipping R2R environments with insufficient instructions). This suggests that the language is as diverse as many of the environments in the original dataset. Qualitatively, the instructions mention a variety of objects (e.g., water fountain, art, sectional sofa), along with attributes such as color (12 mentioned), state (e.g., open, closed, hanging), size (e.g., small, big), and composition (e.g., wood, glass, cement) using both allocentric and egocentric reference frames.

Robot Platform We conduct experiments using a low-cost TurtleBot2 robot consisting of a Kobuki mobile base, an Asus Xion Pro Live RGBD camera, and an Asus netbook. Since virtually all VLN agents have been developed using 360° vision, we additionally equip the robot with a Ricoh Theta V 360° consumer-grade RGB camera. For the most direct sim-to-real comparison we mount the Theta V camera 1.35m above ground level – the same height we set the Matterport camera tripod when scanning and reconstructing Coda. Lastly, for obstacle avoidance and mapping we mount a 270° Hokuyo 2D laser scanner with 30m range at 0.24m above ground level. The robot runs ROS-kinetic [22]. We use standard ROS / TurtleBot packages such as gmapping, amcl and move_base as well as PyTorch [46]. During evaluation all code executes on the robot, except PyTorch ROS nodes (the VLN agent and our subgoal model) which are run remotely.

Evaluation Metrics We use standard VLN metrics for evaluation in both sim and real, with the aim of testing whether performance on the robot can match the results in simulation for the same unseen test environment. In all experiments the agent must terminate the episode as near to the goal position as possible. An episode is considered successful if the navigation error (defined as the distance between the agent's final position and the goal position) is less than 3m. We report average values for trajectory length (TL), navigation error (NE), success rate at reaching the goal (SR), oracle success rate (OS) – defined as the agent's success rate at the closest point on the agent's trajectory to the goal, Success weighted by (normalized inverse) Path Length (SPL) [47] and both Normalized and Success-weighted Dynamic Time Warping (NDTW and SDTW) [48]. SPL and SDTW are summary measures of navigation performance that balance success against trajectory efficiency and fidelity (i.e., similarity to the ground-truth path), while NDTW measures path fidelity irrespective of success (higher is better for each). When calculating NDTW and SDTW we represent both sim and robot trajectories using 100 equally spaced points to eliminate sampling differences.

Robot Pose Tracking To compute these evaluation metrics we must know the agent's pose at all points in time. In the simulation this is available. To track the robot's pose, we first teleop the robot through Coda to construct a 0.05m resolution occupancy map using the laser scanner and the ROS gmapping SLAM package. We then register this map to the Matterport3D coordinate frame. For evaluation purposes we use the particle filter provided by the ROS amcl package to track the robot's pose during experiments. We estimate that the error in this pose estimate is typically an order of magnitude less than the 3m radius used for determining success.

4 Adapting a VLN Agent to a Physical Robot

VLN Agent We now detail our approach to sim-to-real transfer of a VLN agent onto the robot. We start with an existing VLN agent [19] that achieved state-of-the-art performance on the unseen environments in the R2R test split. Typical of most recent work on this task, this agent is trained using a (simulated) panoramic 360° RGB sensor, processing the entire visual context at each step, and a highly-abstracted discrete action space provided by the Matterport3D simulator. Actions are defined by neighboring panoramic image viewpoints, which are on average 2.1m away. The agent processes the visual representations in the direction of those locations, and then once an action has been selected the simulator teleports the agent to the new viewpoint (refer Figure 2). To transfer this agent to a robot we must address differences in the visual domain and action space, plus navigation and localization errors that are not experienced in training. We discuss these challenges in turn.

Visual Domain Adaptation As illustrated by the RGB panorama differences in Figure 2, the robot's Ricoh Theta V camera is consumer-grade with limited dynamic range compared to the Matterport camera, resulting in a loss of visual detail and a 6.4% reduction in success rate due to visual domain
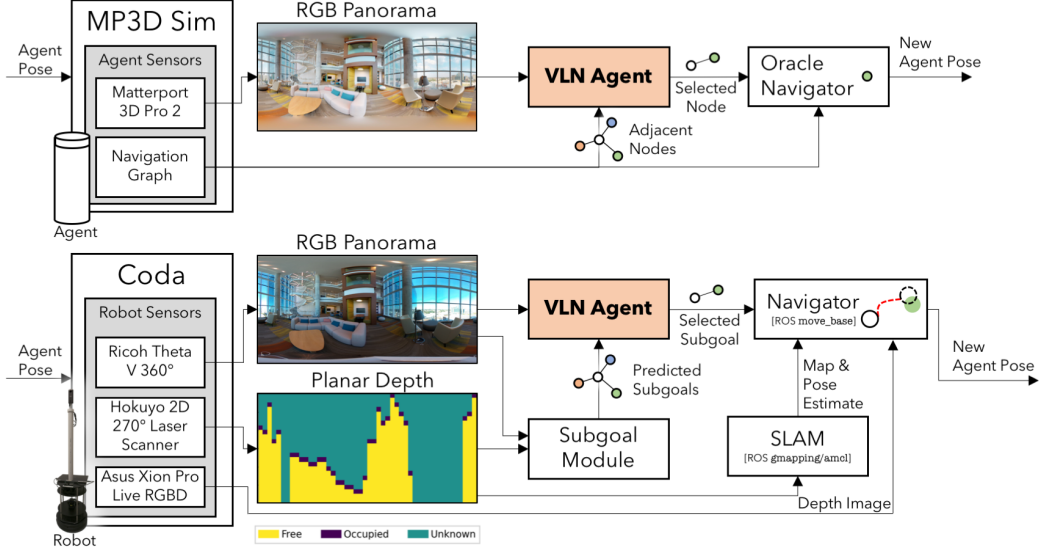
4

Figure 2: We evaluate a VLN agent trained in simulation in matching previously unseen sim (top) and real (bottom) environments. Our sim-to-real framework includes domain randomization, a subgoal model to bridge the action space differences between sim and real (removing the reliance on the simulator navigation graph), and standard ROS components for SLAM and navigation.

differences (refer Section 5). To address this problem, domain adaptation algorithms typically require a large set of target domain images – in this case Theta V panoramic images captured in a variety of indoor environments – which are non-trivial to collect. In preliminary experiments, simple alternatives such as histogram matching to align the Theta V colorspace with the Matterport camera were not effective. We therefore selected an approach akin to domain randomization [24], in which we applied random color jitter to each panorama when training the VLN agent in the Matterport simulator. Brightness, contrast and saturation were varied by a factor of 0.3 and hue was varied by a factor of 0.01, with these parameters determined through visual inspection.

Action Space Adaptation To accommodate the transfer from the highly-abstracted action space of the simulator, which is based on a navigation graph, to the primitive motor control actions of a physical robot, we propose to use a *subgoal model* in conjunction with standard ROS navigation modules. After conditioning on available sensor observations, the subgoal model predicts a set of nearby waypoints, or subgoals, for the VLN agent to choose from. In effect, we provide the VLN agent with an implementation of the simulator's action space and depend on ROS to execute those actions. Our approach is therefore a modular fusion of both classical and learning-based methods.

As illustrated in the example in Figure 3 left, as input to the subgoal model we use the most recent laser scan, to provide an approximate indication of free space, and visual inputs to fill in the gaps and add additional context. We represent the laser scan as a radial occupancy map over range and heading bins, and the visual input using pretrained ResNet-152 [50] CNN features captured at 12 different headings and 3 different camera evaluations (the same visual feature representation consumed by the VLN agent). The subgoal model is based on a 4-stage U-Net [49] architecture that takes the radial occupancy map as input and fuses the visual features in the 3rd downsampling stage. The output of the model is the probability that each laser scan heading and range bin contains a waypoint.

The subgoal model is trained and validated on viewpoints from the Matterport3D dataset [21]. Since the dataset does not include planar laser scans, we simulate the output of a 270° laser scan at each viewpoint by measuring the range to the reconstructed matterport mesh. The model is trained to minimize the sinkhorn divergence [51] between predicted waypoint locations and the location of neighboring viewpoints in the navigation graph, after removing neighboring viewpoints that would require traversing stairs. Sinkhorn divergence is a smoothed approximation to earth mover's distance that we find to be more effective than cross-entropy, which does not respect the underlying metric space. At test time in the Coda environment we use a confidence threshold to select the final set of
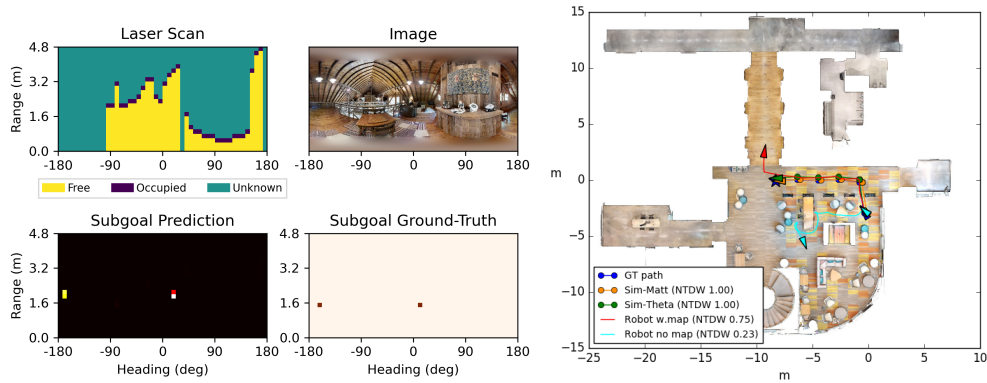
Figure 3: Left: To predict the next set of potential waypoints, or subgoals, we combine a radial occupancy map representation of the laser scan (top-left) with pretrained CNN features from the panoramic image (top-middle) in a U-Net [49] architecture. Even with missing range data (green regions) due to the 270° scan, the model generally predicts plausible subgoals learned from the Matterport simulator navigation graphs (bottom-left vs. bottom-middle). However, when the subgoal model fails to predict a valid waypoint (right, where the subgoal model does not predict a waypoint passing between narrow bookshelves) then the robot navigation fails.

waypoints (up to a maximum of 5) that are provided to the VLN agent. Using the Theta V camera and evaluating at each viewpoint in the Coda navigation graph we find that 29% of predicted waypoints are within 0.5m of a ground-truth neighboring viewpoint (60% within 1m, 74% within 1.5m).

Obstacle Avoidance and Path Planning For obstacle detection we rely on the depth camera and the 270° laser scanner. These sensors are complimentary since, although the camera only has a narrow horizontal field of view, it senses overhanging obstacles such as table tops that may be missed in the narrow planar sweep of the laser scanner (and could be impacted by our tall robot). We rely on the ROS move_base navigation module to integrate these observations, maintain local and global costs maps, and to plan and execute motion to the waypoint selected by the VLN model.

Verification Porting an existing VLN agent to ROS required significant refactoring of the original codebase. To verify our implementation we created mock ROS nodes for the robot's panoramic camera, the subgoal model, and the move_base navigation package using images from the simulator and the simulator navigation graph. This allowed us to run our robot code on simulator data and verified that the performance matched the original R2R-EnvDrop model [19] within 1%. Given the large number of different VLN agents and architectures that have been proposed [14–20], we hope that our framework will ease the sim-to-real burden in future work. As shown in Figure 2, this codebase acts as a ROS-based harness around the standard VLN agent api adopted by the community.

5 Results and Analysis

In this section, we report sim-to-real results and analysis for the VLN robot. We first characterize the difficulty of the Coda environment relative to R2R, and examine the importance of visual domain adaptation. We then consider two sim-to-real settings: the first in which the robot is provided with a laser-scanned occupancy map of the environment and the simulator navigation graph ('with map'), and the second in which the environment is completely unseen at the beginning of each episode, i.e., there is no provided map or navigation graph and the robot's SLAM map is reset each time an instruction is received ('no map'). No aspect of our system is trained or validated in the Coda environment (in simulation or in reality) in either setting. All experiments are conducted in daylight and furniture is restored to its original position before each evaluation. This does not assist the robot (which, in our hardest setting, experiences the environment as previously unseen each time an instruction is received). Rather, this minimizes drift between the simulator and the physical environment, enabling us to more accurately characterize sim-to-real performance.

How challenging is Coda? We first establish the relative difficulty of the Coda environment compared to existing environments in the Matterport3D [21] / R2R [13] dataset. We report the performance of the R2R-EnvDrop [19] VLN agent in the Coda simulator compared to R2R val and test. As illustrated in Table 1, after training on the R2R training set we achieved a 49.9% success rate on the R2R val-unseen set, and a 49.2% success rate on R2R test. On Coda (sim), the agent's success rate is slightly higher at 55.9%. This suggests that, although the Coda dataset was collected by a different camera operator, and a different pool of AMT workers at a different time to R2R, the collection protocol has been faithfully followed and the dataset is 'in-domain' with respect to R2R. Accordingly, we treat Coda sim performance (row 4) as a baseline to investigate sim-to-real transfer.

	Split	TL (m)	NE (m)	OS (%)	SR (%)	SPL	SDTW	NDTW
1	R2R Val-Seen	9.70	4.56	62.0	57.5	55.0	49.6	60.8
2	R2R Val-Unseen	9.18	5.42	55.4	49.9	47.0	44.5	55.2
3	R2R Test	9.52	5.57	54.9	49.2	46.8	-	-
4	Coda	11.22	4.98	59.5	55.9	53.6	44.0	54.9

Table 1: Characterizing the difficulty of the Coda simulator environment using the EnvDrop agent [19]. Performance in Coda is slightly higher than the R2R test set on average. R2R Val-Unseen, R2R Test and Coda are previously unseen visual environments.

How important is visual domain adaptation? To address this question, we replaced the Matterport panoramic images in the Coda simulator using images captured with the robot's Ricoh Theta V camera from the same viewpoint locations. This allows us to isolate the importance of visual domain adaptation in the absence of other factors. As foreshadowed in Figure 2, switching to the cheaper camera (row 5 vs. row 4 in Table 2) causes a 6.4% drop in success rate and a 6.2% drop in SPL, which is reduced to a 3.9% and 4.3% drop respectively (row 6 vs. row 4) when we retrained the agent using visual domain adaptation. To make these results as reliable as possible, rows 5 and 6 are averaged over 3 sets of Theta V images that were captured on different days, exhibiting a variation in success rate of ±1.9% without domain adaptation and ±1.4% with domain adaptation.

	Setting	Camera	Adapted	Map	TL (m)	NE (m)	OS (%)	SR (%)	SPL	SDTW	NDTW
4	Sim	Matterport	-	-	11.22	4.98	59.5	55.9	53.6	44.0	54.9
5	Sim	Theta V	✗	-	11.21	5.71	55.3	49.5	47.4	39.5	52.0
6	Sim	Theta V	✓	-	11.38	5.74	59.2	52.0	49.3	42.6	54.0
7	Robot	Theta V	✓	✓	11.32	6.04	51.4	46.8	43.9	28.7	38.5
8	Robot	Theta V	✓	✗	8.02	6.56	26.1	22.5	21.9	13.8	30.0

Table 2: Sim-to-real performance comparison for our VLN agent in Coda over 1,334m of language-guided trajectories. Success rate at reaching the goal (SR) and path fidelity (NDTW) remains relatively high in the robot 'with map' setting (row 7), but is reduced in the 'no map' setting (row 8).

How large is the sim-to-real gap with a map? In the 'with map' setting, we conduct a full evaluation on the robot while also providing a pre-captured laser scan to assist with obstacle avoidance and path-planning, and the simulator navigation graph to provide waypoint candidates. The subgoal model is not used. We consider this setting for two reasons. First, for a robot operating in a single environment it may be reasonable to provide some navigation annotations and to expect the robot to maintain an occupancy map. Second, this setting tests the implicit assumption in graph-based simulators that existing robotics systems are capable of navigating between viewpoints in the graph.

As reported in Table 2, in the 'with map' setting the robot achieves an instruction-following success rate of 46.8% in physical Coda, a reduction of 5.2% compared to using the robot camera in simulation (row 7 vs row 6). During this evaluation over 111 instructions and 1.3km of trajectories we did not experience any collisions or navigation failures, which we define as the robot failing to navigate an edge between panoramic viewpoints in the navigation graph. For context, the environment contains five 'squeeze points' with 80-90cm gaps between furniture, compared to which the width of the robot with laptop is around 40cm. We attribute the reduction in performance to a degradation of visual inputs from viewpoints that are slightly out-of-position compared to the simulator. We also note that,

7

despite our best efforts, there were some people present in Coda at times during testing, which may also degrade robot performance due to domain differences (images of people are extremely rare in the Matterport3D dataset). Overall we conclude that, at least based on analysis performed in the Coda environment, if an occupancy map and navigation graph are collected and annotated in advance, transferring a VLN agent to an inexpensive robot using a classical navigation stack is feasible with around a 9.1% reduction in success rate (row 7 vs. row 4).

How large is the sim-to-real gap overall? In the final experimental setting, we revoke the robot's access to the Coda occupancy map and the simulator navigation graph. Removing the navigation graph requires the robot to rely on waypoint predictions from the subgoal model for the first time. Removing the occupancy map requires the robot to perform simultaneous localization and mapping (SLAM) from scratch in each episode (every time a new instruction is received), which makes obstacle avoidance and path planning more challenging.[3] Overall, this setting is indicative of 'cold-start' performance in a new environment (which is also the standard VLN evaluation setting in simulation).

As reported in row 8 of Table 2, in the 'no map' setting the instruction-following success rate drops a further 24.3% to 22.5%. This experiment also exhibits the lowest trajectory similarity with the results of the Matterport3D simulator (Table 3). Without the occupancy map, the robot collided with objects (a table and a chair) for the first time. However, we attribute most of the drop in success rate to differences between the predictions of the subgoal model and the locations of viewpoints in the simulator navigation graph. We found that both false positive and false negative waypoint predictions were evident, e.g., when the robot attempted to navigate down the spiral staircase (false positive), or failed to consider navigating through the narrow 'squeeze points' between bookshelves and other furniture (false negatives). Based on these results, we consider VLN sim-to-real transfer in this setting to be an open research problem.

	Platform	Camera	Adaptation	Map	4	5	6	7	8
4	Sim	Matterport	-	-	1.00	0.78	0.66	0.41	0.33
5	Sim	Theta V	✗	-	0.78	1.00	0.70	0.42	0.33
6	Sim	Theta V	✓	-	0.66	0.70	1.00	0.44	0.33
7	Robot	Theta V	✓	✓	0.41	0.42	0.44	1.00	0.57
8	Robot	Theta V	✓	✗	0.31	0.33	0.33	0.57	1.00

Table 3: Trajectory similarity between Coda experimental settings based on Normalized Dynamic Time Warping (NDTW) [48]. Figure 3 right provides an example to contextualize NDTW values.

6 Conclusion and Future Directions

We attempt the first sim-to-real transfer of a Vision-and-Language Navigation (VLN) agent trained on the R2R dataset to a low-cost robot with 360° vision, using a learned subgoal model and classical SLAM and path-planning routines. We show that, if an occupancy map and navigation graph can be collected and annotated in advance, sim-to-real transfer is largely successful albeit with a ~9% reduction in instruction following success due to visual domain differences (~4%) and viewpoint differences (~5%). This is a promising result, suggesting that the language groundings learned by the agent in simulation can transfer to a physical environment not seen in training. However, in the hardest 'cold start' setting with no prior mapping of the environment, sim-to-real transfer is much less reliable due to the failure of the subgoal model to predict the same neighboring waypoints in the simulator navigation graph. To narrow the sim-to-real gap in future work, the subgoal model will need to be improved or eliminated, perhaps by training the VLN agent using a low-level action space (e.g., predicting the heading and distance to move). Since the graph-based Matterport3D simulator cannot support these low-level actions, this would require off-policy reinforcement learning algorithms that can learn from a fixed batch of data that has already been gathered [52], or alternatively, switching to a simulator that supports continuous motion [53, 54], as in recent work from Krantz et al. [55]. Since these simulators introduce visual artifacts not present in the graph-based simulator, this may exacerbate visual domain differences, although recent work [56] demonstrates a promisingly small sim-to-real gap in the context of PointGoal navigation [57].

---

[3]Note that for pose tracking, we run a separate, isolated ROS navigation stack with map access.

References

[1] S. Guadarrama, L. Riano, D. Golland, D. Go, Y. Jia, D. Klein, P. Abbeel, T. Darrell, et al. Grounding spatial relations for human-robot interaction. In *IROS*, 2013.

[2] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35(1-3):281–300, 2016.

[3] R. Paul, J. Arkin, N. Roy, and T. M Howard. Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators. *RSS*, 2016.

[4] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *AAAI*, 2011.

[5] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011.

[6] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pages 403–415. Springer, 2013.

[7] Y. Bisk, D. Yuret, and D. Marcu. Natural language communication with robots. In *NAACL-HLT*, 2016.

[8] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *arXiv:1706.07230*, 2017.

[9] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *Human-Robot Interaction (HRI)*, 2010.

[10] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *AAAI*, 2006.

[11] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, 2016.

[12] A. Vogel and D. Jurafsky. Learning to follow navigational directions. In *ACL*, 2010.

[13] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018.

[14] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019.

[15] X. Wang, W. Xiong, H. Wang, and W. Y. Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *ECCV*, 2018.

[16] C.-Y. Ma, J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher, and C. Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019.

[17] C.-Y. Ma, Z. Wu, G. AlRegib, C. Xiong, and Z. Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *CVPR*, 2019.

[18] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018.

[19] H. Tan, L. Yu, and M. Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL*, 2019.

[20] L. Ke, X. Li, Y. Bisk, A. Holtzman, Z. Gan, J. Liu, J. Gao, Y. Choi, and S. Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *CVPR*, 2019.

[21] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[23] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 2007.

[24] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

[25] S. Hemachandra, F. Duvallet, T. M. Howard, N. Roy, A. Stentz, and M. R. Walter. Learning models for following natural language directions in unknown environments. In *ICRA*, 2015.

[26] A. S. Huang, S. Tellex, A. Bachrach, T. Kollar, D. Roy, and N. Roy. Natural language command of an autonomous micro-air vehicle. In *IROS*, 2010.

[27] V. Blukis, Y. Terme, E. Niklasson, R. A. Knepper, and Y. Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *CoRL*, 2019.

[28] J. Oh, T. M. Howard, M. R. Walter, D. Barber, M. Zhu, S. Park, A. Suppe, L. Navarro-Serment, F. Duvallet, A. Boularias, et al. Integrated intelligence for human-robot teams. In *International Symposium on Experimental Robotics*, pages 309–322. Springer, 2016.

[29] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *ICRA*, 2018.

[30] S. Patki, E. Fahnestock, T. M. Howard, and M. R. Walter. Language-guided semantic mapping and mobile manipulation in partially observable environments. In *CoRL*, 2019.

[31] M. Tucker, D. Aksaray, R. Paul, G. J. Stein, and N. Roy. Learning unknown groundings for natural language interaction with mobile robots. In *Robotics Research*, pages 317–333. Springer, 2020.

[32] S. Banerjee, J. Thomason, and J. J. Corso. The RobotSlang Benchmark: Dialog-guided robot localization and navigation. In *CORL*, 2020.

[33] E. Fahnestock, S. Patki, and T. M. Howard. Language-guided adaptive perception with hierarchical symbolic representations for mobile manipulators. *arXiv:1909.09880*, 2019.

[34] R. Paul, A. Barbu, S. Felshin, B. Katz, and N. Roy. Temporal grounding graphs for language understanding with accrued visual-linguistic context. *arXiv:1811.06966*, 2018.

[35] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer. Vision-and-dialog navigation. In *CoRL*, 2019.

[36] K. Nguyen and H. Daumé III. Help, Anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *EMNLP*, 2019.

[37] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen, and A. van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *CVPR*, 2020.

[38] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *EMNLP*, 2020.

[39] A. Yan, X. Wang, J. Feng, L. Li, and W. Y. Wang. Cross-lingual vision-language navigation. *arXiv:1910.11301*, 2019.

[40] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell, et al. Learning to navigate in cities without a map. In *NeurIPS*, 2018.

[41] H. Mehta, Y. Artzi, J. Baldridge, E. Ie, and P. Mirowski. Retouchdown: Adding touchdown to streetlearn as a shareable resource for language grounding tasks in street view. *EMNLP Workshop on Spatial Language Understanding (SpLU)*, 2020.

[42] M. Weiss, S. Chamorro, R. Girgis, M. Luck, S. E. Kahou, J. P. Cohen, D. Nowrouzezahrai, D. Precup, F. Golemo, and C. Pal. Navigation agents for the visually impaired: A sidewalk simulator and experiments. *arXiv:1910.13249*, 2019.

[43] H. Chen, A. Suhr, D. Misra, and Y. Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*, 2019.

[44] A. B. Vasudevan, D. Dai, and L. Van Gool. Talk2nav: Long-range vision-and-language navigation in cities. *arXiv:1910.02029*, 2019.

[45] T. Paz-Argaman and R. Tsarfaty. Run through the streets: A new dataset and baseline models for realistic urban navigation. *arXiv:1909.08970*, 2019.

[46] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[47] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir. On evaluation of embodied navigation agents. *arXiv:1807.06757*, 2018.

[48] G. Magalhaes, V. Jain, A. Ku, E. Ie, and J. Baldridge. Effective and general evaluation for instruction conditioned navigation using dynamic time warping. *arXiv:1907.05446*, 2019.

[49] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[50] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[51] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, 2013.

[52] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

[53] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. Gibson env: real-world perception for embodied agents. In *CVPR*, 2018.

[54] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *ICCV*, 2019.

[55] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *ECCV*, 2020.

[56] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Are we making real progress in simulated environments? Measuring the sim2real gap in embodied visual navigation. In *IROS*, 2020.

[57] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv:1807.06757*, 2018.