Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments

Peter Anderson[1]   Qi Wu[2]   Damien Teney[2]   Jake Bruce[3]   Mark Johnson[4]

Niko Sünderhauf[3]   Ian Reid[2]   Stephen Gould[1]   Anton van den Hengel[2]

[1]Australian National University [2]University of Adelaide [3]Queensland University of Technology [4]Macquarie University

[1]firstname.lastname@anu.edu.au, [3]jacob.bruce@hdr.qut.edu.au, [3]niko.suenderhauf@qut.edu.au

[2]{qi.wu01,damien.teney,ian.reid,anton.vandenhengel}@adelaide.edu.au, [4]mark.johnson@mq.edu.au

Abstract

A robot that can carry out a natural-language instruction has been a dream since before the Jetsons cartoon series imagined a life of leisure mediated by a fleet of attentive robot helpers. It is a dream that remains stubbornly distant. However, recent advances in vision and language methods have made incredible progress in closely related areas. This is significant because a robot interpreting a natural-language navigation instruction on the basis of what it sees is carrying out a vision and language process that is similar to Visual Question Answering. Both tasks can be interpreted as visually grounded sequence-to-sequence translation problems, and many of the same methods are applicable. To enable and encourage the application of vision and language methods to the problem of interpreting visually-grounded navigation instructions, we present the Matterport3D Simulator – a large-scale reinforcement learning environment based on real imagery [11]. Using this simulator, which can in future support a range of embodied vision and language tasks, we provide the first benchmark dataset for visually-grounded natural language navigation in real buildings – the Room-to-Room (R2R) dataset[1].

1. Introduction

The idea that we might be able to give general, verbal instructions to a robot and have at least a reasonable probability that it will carry out the required task is one of the long-held goals of robotics, and artificial intelligence (AI). Despite significant progress, there are a number of major technical challenges that need to be overcome before robots will be able to perform general tasks in the real world. One of the primary requirements will be new techniques for linking natural language to vision and action in *unstructured, previously unseen environments*. It is the navigation version
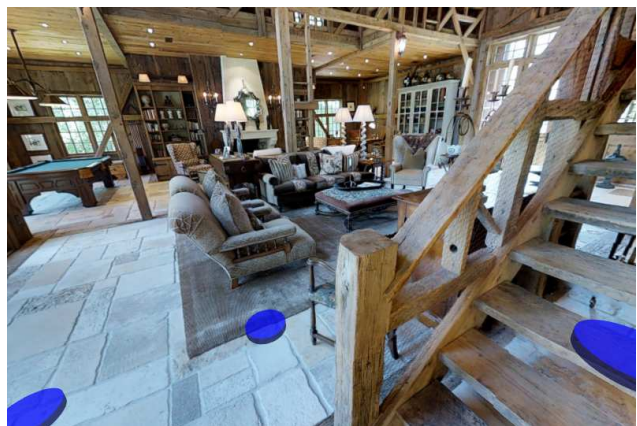
Instruction: Head upstairs and walk past the piano through an archway directly in front. Turn right when the hallway ends at pictures and table. Wait by the moose antlers hanging on the wall.

Figure 1.   Room-to-Room (R2R) navigation task. We focus on executing natural language navigation instructions in previously unseen real-world buildings. The agent's camera can be rotated freely. Blue discs indicate nearby (discretized) navigation options.

of this challenge that we refer to as Vision-and-Language Navigation (VLN).

Although interpreting natural-language navigation instructions has received significant attention previously [12, 13, 20, 38, 41, 52], it is the recent success of recurrent neural network methods for the joint interpretation of images and natural language that motivates the VLN task, and the associated Room-to-Room (R2R) dataset described below. The dataset particularly has been designed to simplify the application of vision and language methods to what might otherwise seem a distant problem.

Previous approaches to natural language command of robots have often neglected the visual information processing aspect of the problem. Using rendered, rather than real images [7, 27, 62], for example, constrains the set of vis-
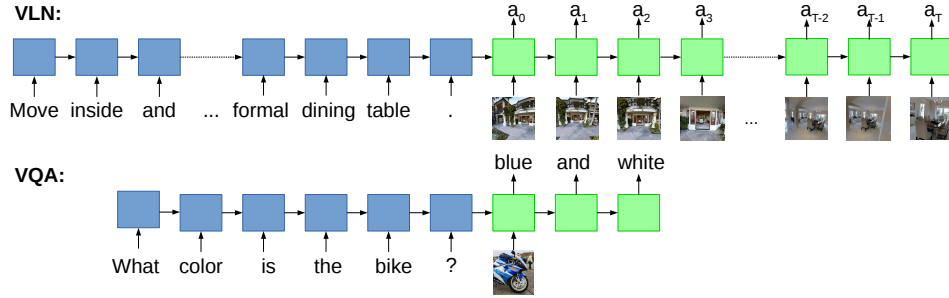
---

[1]https://bringmeaspoon.org

Figure 2. Differences between Vision-and-Language Navigation (VLN) and Visual Question Answering (VQA). Both tasks can be formulated as visually grounded sequence-to-sequence transcoding problems. However, VLN sequences are much longer and, uniquely among vision and language benchmark tasks using real images, the model outputs actions $\langle a_0, a_1, \ldots a_T \rangle$ that manipulate the camera viewpoint.

ible objects to the set of hand-crafted models available to the renderer. This turns the robot's challenging open-set problem of relating real language to real imagery into a far simpler closed-set classification problem. The natural extension of this process is that adopted in works where the images are replaced by a set of labels [13, 52]. Limiting the variation in the imagery inevitably limits the variation in the navigation instructions also. What distinguishes the VLN challenge is that the agent is required to interpret a previously *unseen* natural-language navigation command in light of images generated by a previously *unseen* real environment. The task thus more closely models the distinctly open-set nature of the underlying problem.

To enable the reproducible evaluation of VLN methods, we present the Matterport3D Simulator. The simulator is a large-scale interactive reinforcement learning (RL) environment constructed from the Matterport3D dataset [11] which contains 10,800 densely-sampled panoramic RGB-D images of 90 real-world building-scale indoor environments. Compared to synthetic RL environments [7, 27, 62], the use of real-world image data preserves visual and linguistic richness, maximizing the potential for trained agents to be transferred to real-world applications.

Based on the Matterport3D environments, we collect the Room-to-Room (R2R) dataset containing 21,567 open-vocabulary, crowd-sourced navigation instructions with an average length of 29 words. Each instruction describes a trajectory traversing typically multiple rooms. As illustrated in Figure 1, the associated task requires an agent to follow natural-language instructions to navigate to a goal location in a previously unseen building. We investigate the difficulty of this task, and particularly the difficulty of operating in unseen environments, using several baselines and a sequence-to-sequence model based on methods successfully applied to other vision and language tasks [4, 14, 19].

In summary, our main contributions are:

1. We introduce the Matterport3D Simulator, a software framework for visual reinforcement learning using the Matterport3D panoramic RGB-D dataset [11];
2. We present Room-to-Room (R2R), the first benchmark dataset for Vision-and-Language Navigation in real, previously unseen, building-scale 3D environments;
3. We apply sequence-to-sequence neural networks to the R2R dataset, establishing several baselines.

The simulator, R2R dataset and baseline models are available through the project website at https://bringmeaspoon.org.

2. Related Work

Navigation and language Natural language command of robots in unstructured environments has been a research goal for several decades [57]. However, many existing approaches abstract away the problem of visual perception to some degree. This is typically achieved either by assuming that the set of all navigation goals, or objects to be acted upon, has been enumerated, and that each will be identified by label [13, 52], or by operating in visually restricted environments requiring limited perception [12, 20, 24, 29, 35, 38, 55]. Our work contributes for the first time a navigation benchmark dataset that is both linguistically and visually rich, moving closer to real scenarios while still enabling reproducible evaluations.

Vision and language The development of new benchmark datasets for image captioning [14], visual question answering (VQA) [4, 19] and visual dialog [17] has spurred considerable progress in vision and language understanding, enabling models to be trained end-to-end on raw pixel data from large datasets of natural images. However, although many tasks combining visual and linguistic reasoning have been motivated by their potential robotic applications [4, 17, 26, 36, 51], none of these tasks allow an agent to move or control the camera. As illustrated in Figure 2, our proposed R2R benchmark addresses this limitation, which also motivates several concurrent works on embodied question answering [16, 18].

Navigation based simulators Our simulator is related to existing 3D RL environments based on game engines, such as ViZDoom [27], DeepMind Lab [7] and AI2-THOR [30], as well as a number of newer environments developed concurrently including HoME [10], House3D [58], MINOS [47], CHALET [59] and Gibson Env [61]. The main advantage of our framework over synthetic environments [30, 10, 58, 59] is that all pixel observations come from natural images of real scenes, ensuring that almost every coffee mug, pot-plant and wallpaper texture is unique. This visual diversity and richness is hard to replicate using a limited set of 3D assets and textures. Compared to MINOS [47], which is also based on Matterport data [11], we render from panoramic images rather than textured meshes. Since the meshes have missing geometry – particularly for windows and mirrors – our approach improves visual realism but limits navigation to discrete locations (refer Section 3.2 for details). Our approach is similar to the (much smaller) Active Vision Dataset [2].

RL in navigation A number of recent papers use reinforcement learning (RL) to train navigational agents [31, 50, 53, 62, 21], although these works do not address language instruction. The use of RL for language-based navigation has been studied in [12] and [41], however, the settings are visually and linguistically less complex. For example, Chaplot *et al.* [12] develop an RL model to execute template-based instructions in Doom environments [27]. Misra *et al.* [41] study complex language instructions in a fully-observable blocks world. By releasing our simulator and dataset, we hope to encourage further research in more realistic partially-observable settings.

3. Matterport3D Simulator

In this section we introduce the Matterport3D Simulator, a new large-scale visual reinforcement learning (RL) simulation environment for the research and development of intelligent agents based on the Matterport3D dataset [11]. The Room-to-Room (R2R) navigation dataset is discussed in Section 4.

3.1. Matterport3D Dataset

Most RGB-D datasets are derived from video sequences; e.g. NYUv2 [42], SUN RGB-D [48] and ScanNet [15]. These datasets typically offer only one or two paths through a scene, making them inadequate for simulating robot motion. In contrast to these datasets, the recently released Matterport3D dataset [11] contains a comprehensive set of panoramic views. To the best of our knowledge it is also the largest currently available RGB-D research dataset.

In detail, the Matterport3D dataset consists of 10,800 panoramic views constructed from 194,400 RGB-D images of 90 building-scale scenes. On average, panoramic view-points are distributed throughout the entire walkable floor plan of each scene at an average separation of 2.25m. Each panoramic view is comprised of 18 RGB-D images captured from a single 3D position at the approximate height of a standing person. Each image is annotated with an accurate 6 DoF camera pose, and collectively the images capture the entire sphere except the poles. The dataset also includes globally-aligned, textured 3D meshes annotated with class and instance segmentations of regions (rooms) and objects.

In terms of visual diversity, the selected Matterport scenes encompass a range of buildings including houses, apartments, hotels, offices and churches of varying size and complexity. These buildings contain enormous visual diversity, posing real challenges to computer vision. Many of the scenes in the dataset can be viewed in the Matterport 3D spaces gallery[2].

3.2. Simulator

3.2.1 Observations

To construct the simulator, we allow an embodied agent to virtually 'move' throughout a scene by adopting poses coinciding with panoramic viewpoints. Agent poses are defined in terms of 3D position $v \in V$, heading $\psi \in [0, 2\pi)$, and camera elevation $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, where V is the set of 3D points associated with panoramic viewpoints in the scene. At each step t, the simulator outputs an RGB image observation o_t corresponding to the agent's first person camera view. Images are generated from perspective projections of precomputed cube-mapped images at each viewpoint. Future extensions to the simulator will also support depth image observations (RGB-D), and additional instrumentation in the form of rendered object class and object instance segmentations (based on the underlying Matterport3D mesh annotations).

3.2.2 Action Space

The main challenge in implementing the simulator is determining the state-dependent action space. Naturally, we wish to prevent agents from teleporting through walls and floors, or traversing other non-navigable regions of space. Therefore, at each step t the simulator also outputs a set of next step reachable viewpoints $W_{t+1} \subseteq V$. Agents interact with the simulator by selecting a new viewpoint $v_{t+1} \in W_{t+1}$, and nominating camera heading ($\Delta\psi_{t+1}$) and elevation ($\Delta\theta_{t+1}$) adjustments. Actions are deterministic.

To determine W_{t+1}, for each scene the simulator includes a weighted, undirected graph over panoramic viewpoints, $G = \langle V, E \rangle$, such that the presence of an edge signifies a robot-navigable transition between two viewpoints,

---

[2]https://matterport.com/gallery/

and the weight of that edge reflects the straight-line distance between them. To construct the graphs, we ray-traced between viewpoints in the Matterport3D scene meshes to detect intervening obstacles. To ensure that motion remains localized, we then removed edges longer than 5m. Finally, we manually verified each navigation graph to correct for missing obstacles not captured in the meshes (such as windows and mirrors).

Given navigation graph G, the set of next-step reachable viewpoints is given by:

$$W_{t+1} = \{v_t\} \cup \{v_i \in V \mid \langle v_t, v_i \rangle \in E \wedge v_i \in P_t\} \quad (1)$$

where v_t is the current viewpoint, and P_t is the region of space enclosed by the left and right extents of the camera view frustum at step t. In effect, the agent is permitted to follow any edges in the navigation graph, provided that the destination is within the current field of view, or visible by glancing up or down[3]. Alternatively, the agent always has the choice to remain at the same viewpoint and simply move the camera.

Figure 3 illustrates a partial example of a typical navigation graph. On average each graph contains 117 viewpoints, with an average vertex degree of 4.1. This compares favorably with grid-world navigation graphs which, due to walls and obstacles, must have an average degree of less than 4. As such, although agent motion is discretized, this does not constitute a significant limitation in the context of most high-level tasks. Even with a real robot it may not be practical or necessary to continuously re-plan higher-level objectives with every new RGB-D camera view. Indeed, even agents operating in 3D simulators that notionally support continuous motion typically use discretized action spaces in practice [62, 16, 18, 47].

The simulator does not define or place restrictions on the agent's goal, reward function, or any additional context (such as natural language navigation instructions). These aspects of the RL environment are task and dataset dependent, for example as described in Section 4.

3.2.3 Implementation Details

The Matterport3D Simulator is written in C++ using OpenGL. In addition to the C++ API, Python bindings are also provided, allowing the simulator to be easily used with deep learning frameworks such as Caffe [25] and Tensor-Flow [1], or within RL platforms such as ParlAI [39] and OpenAI Gym [9]. Various configuration options are offered for parameters such as image resolution and field of view. Separate to the simulator, we have also developed a WebGL browser-based visualization library for collecting text annotations of navigation trajectories using Amazon Mechanical Turk, which we will make available to other researchers.

---

[3]This avoids forcing the agent to look at the floor every time it takes a small step.
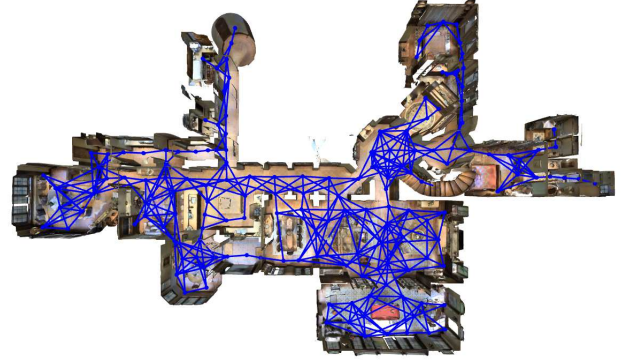

Figure 3. Example navigation graph for a partial floor of one building-scale scene in the Matterport3D Simulator. Navigable paths between panoramic viewpoints are illustrated in blue. Stairs can also be navigated to move between floors.

3.2.4 Biases

We are reluctant to introduce a new dataset (or simulator, in this case) without at least some attempt to address its limitations and biases [54]. In the Matterport3D dataset we have observed several selection biases. First, the majority of captured living spaces are scrupulously clean and tidy, and often luxurious. Second, the dataset contains very few people and animals, which are a mainstay of many other vision and language datasets [14, 4]. Finally, we observe some capture bias as selected viewpoints generally offer commanding views of the environment (and are therefore not necessarily in the positions in which a robot might find itself). Alleviating these limitations to some extent, the simulator can be extended by collecting additional building scans. Refer to Stanford 2D-3D-S [5] for a recent example of an academic dataset collected with a Matterport camera.

4. Room-to-Room (R2R) Navigation

We now describe the Room-to-Room (R2R) task and dataset, including an outline of the data collection process and analysis of the navigation instructions gathered.

4.1. Task

As illustrated in Figure 1, the R2R task requires an embodied agent to follow natural language instructions to navigate from a starting pose to a goal location in the Matterport3D Simulator. Formally, at the beginning of each episode the agent is given as input a natural language instruction $\bar{x} = \langle x_1, x_2, \ldots x_L \rangle$, where L is the length of the instruction and x_i is a single word token. The agent observes an initial RGB image o_0, determined by the agent's initial pose comprising a tuple of 3D position, heading and elevation $s_0 = \langle v_0, \psi_0, \theta_0 \rangle$. The agent must execute a sequence of actions $\langle s_0, a_0, s_1, a_1, \ldots, s_T, a_T \rangle$, with each ac-

Pass the pool and go indoors using the double glass doors. Pass the large table with chairs and turn left and wait by the wine bottles that have grapes by them.

Walk straight through the room and exit out the door on the left. Keep going past the large table and turn left. Walk down the hallway and stop when you reach the 2 entry ways. One in front of you and one to your right. The bar area is to your left.

Enter house through double doors, continue straight across dining room, turn left into bar and stop on the circle on the ground.

Standing in front of the family picture, turn left and walk straight through the bathroom past the tub and mirrors. Go through the doorway and stop when the door to the bathroom is on your right and the door to the closet is to your left.

Walk with the family photo on your right. Continue straight into the bathroom. Walk past the bathtub. Stop in the hall between the bathroom and toilet doorways.

Walk straight passed bathtub and stop with closet on the left and toilet on the right.

Exit the office then turn left and then turn left in the hallway and head down the hallway until you get to a door on your left and go into office 359 then stop.

Go out of the room and take a left. Go into the first room on your left.

Leave the office and take a left. Take the next left at the hallway. Walk down the hall and enter the first office on the left. Stop next to the door to office 359.
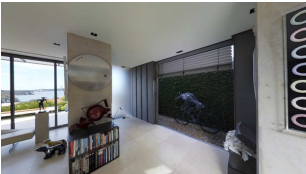
Go up the stairs and turn right. Go past the bathroom and stop next to the bed.

Walk all the way up the stairs, and immediately turn right. Pass the bathroom on the left, and enter the bedroom that is right there, and stop there.

Walk up the stairs turn right at the top and walk through the doorway continue straight and stop inside the bedroom.

Figure 4. Randomly selected examples of navigation instructions (three per trajectory) shown with the view from the starting pose.

tion a_t leading to a new pose $s_{t+1} = \langle v_{t+1}, \psi_{t+1}, \theta_{t+1} \rangle$, and generating a new image observation o_{t+1}. The episode ends when the agent selects the special `stop` action, which is augmented to the simulator action space defined in Section 3.2.2. The task is successfully completed if the action sequence delivers the agent close to an intended goal location v^* (refer to Section 4.4 for evaluation details).

4.2. Data Collection

To generate navigation data, we use the Matterport3D region annotations to sample start pose s_0 and goal location v^* pairs that are (predominantly) in different rooms. For each pair, we find the shortest path $v_0 : v^*$ in the relevant weighted, undirected navigation graph G, discarding paths that are shorter than 5m, and paths that contain less than four or more than six edges. In total we sample 7,189 paths capturing most of the visual diversity in the dataset. The average path length is 10m, as illustrated in Figure 5.

For each path, we collect three associated navigation instructions using Amazon Mechanical Turk (AMT). To this
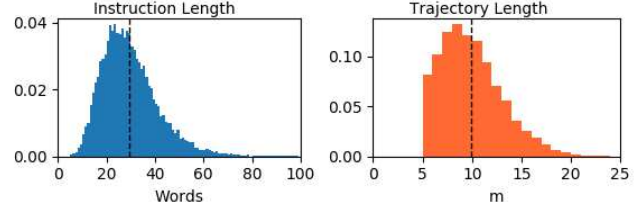

Figure 5. Distribution of instruction length and navigation trajectory length in the R2R dataset.

end, we provide workers with an interactive 3D WebGL environment depicting the path from the start location to the goal location using colored markers. Workers can interact with the trajectory as a 'fly-through', or pan and tilt the camera at any viewpoint along the path for additional context. We then ask workers to 'write directions so that a smart robot can find the goal location after starting from the same start location'. Workers are further instructed that it is not necessary to follow exactly the path indicated, merely to reach the goal. A video demonstration is also provided.

The full collection interface (which is included as supplementary material) was the result of several rounds of experimentation. We used only US-based AMT workers, screened according to their performance on previous tasks. Over 400 workers participated in the data collection, contributing around 1,600 hours of annotation time.

4.3. R2R Dataset Analysis

In total, we collected 21,567 navigation instructions with an average length of 29 words. This is considerably longer than visual question answering datasets where most questions range from four to ten words [4]. However, given the focused nature of the task, the instruction vocabulary is relatively constrained, consisting of around 3.1k words (approximately 1.2k with five or more mentions). As illustrated by the examples included in Figure 4, the level of abstraction in instructions varies widely. This likely reflects differences in people's mental models of the way a 'smart robot' works [43], making the handling of these differences an important aspect of the task. The distribution of navigation instructions based on their first words is depicted in Figure 6. Although we use the R2R dataset in conjunction with the Matterport3D Simulator, we see no technical reason why this dataset couldn't also be used with other simulators based on the Matterport dataset [11].

4.4. Evaluation Protocol

One of the strengths of the R2R task is that, in contrast to many other vision and language tasks such as image captioning and visual dialog, success is clearly measurable. We define *navigation error* as the shortest path distance in the navigation graph G between the agent's final position v_T

Figure 6. Distribution of navigation instructions based on their first four words. Instructions are read from the center outwards. Arc lengths are proportional to the number of instructions containing each word. White areas represent words with individual contributions too small to show.

(i.e., disregarding heading and elevation) and the goal location v^*. We consider an episode to be a *success* if the navigation error is less than 3m. This threshold allows for a margin of error of approximately one viewpoint, yet it is comfortably below the minimum starting error of 5m. We do not evaluate the agent's entire trajectory as many instructions do not specify the path that should be taken.

Central to our evaluation is the requirement for the agent to choose to end the episode when the goal location is identified. We consider stopping to be a fundamental aspect of completing the task, demonstrating understanding, but also freeing the agent to potentially undertake further tasks at the goal. However, we acknowledge that this requirement contrasts with recent works in vision-only navigation that do not train the agent to stop [62, 40]. To disentangle the problem of recognizing the goal location, we also report success for each agent under an *oracle* stopping rule, i.e. if the agent stopped at the closest point to the goal on its trajectory. Misra *et al*. [41] also use this evaluation.

Dataset Splits We follow broadly the same train/val/test split strategy as the Matterport3D dataset [11]. The test set consists of 18 scenes, and 4,173 instructions. We reserve an additional 11 scenes and 2,349 instructions for validating in unseen environments (val unseen). The remaining 61 scenes are pooled together, with instructions split 14,025 train / 1,020 val seen. Following best practice, goal locations for the test set will not be released. Instead, we will provide an evaluation server where agent trajectories may be uploaded for scoring.

5. Vision-and-Language Navigation Agents

In this section, we describe a sequence-to-sequence neural network agent and several other baselines that we use to explore the difficulty of the R2R navigation task.

5.1. Sequence-to-Sequence Model

We model the agent with a recurrent neural network policy using an LSTM-based [23] sequence-to-sequence architecture with an attention mechanism [6]. Recall that the agent begins with a natural language instruction $\bar{x} = \langle x_1, x_2, \ldots x_L \rangle$, and an initial image observation o_0. The encoder computes a representation of $\bar{x}$. At each step t, the decoder observes representations of the current image o_t and the previous action a_{t-1} as input, applies an attention mechanism to the hidden states of the language encoder, and predicts a distribution over the next action a_t. Using this approach, the decoder maintains an internal memory of the agent's entire preceeding history, which is essential for navigating in a partially observable environment [56]. We discuss further details in the following sections.

Language instruction encoding Each word x_i in the language instruction is presented sequentially to the encoder LSTM as an embedding vector. We denote the output of the encoder at step i as h_i, such that $h_i = \text{LSTM}_{enc}(x_i, h_{i-1})$. We denote $\bar{h} = \{h_1, h_2, \ldots, h_L\}$ as the encoder context, which will be used in the attention mechanism. As with Sutskever *et al*. [49], we found it valuable to reverse the order of words in the input language instruction.

Model action space The simulator action space is state-dependent (refer Section 3.2.2), allowing agents to make fine-grained choices between different forward trajectories that are presented. However, in this initial work we simplify our model action space to 6 actions corresponding to `left`, `right`, `up`, `down`, `forward` and `stop`. The `forward` action is defined to always move to the reachable viewpoint that is closest to the centre of the agent's visual field. The `left`, `right`, `up` and `down` actions are defined to move the camera by 30 degrees.

Image and action embedding For each image observation o_t, we use a ResNet-152 [22] CNN pretrained on ImageNet [46] to extract a mean-pooled feature vector. Analogously to the embedding of instruction words, an embedding is learned for each action. The encoded image and previous action features are then concatenated together to form a single vector q_t. The decoder LSTM operates as $h'_t = \text{LSTM}_{dec}(q_t, h'_{t-1})$.

Action prediction with attention mechanism To predict a distribution over actions at step t, we first use an attention mechanism to identify the most relevant parts of the navigation instruction. This is achieved by using the global, general alignment function described by Luong *et al*. [34]

to compute an instruction context $c_t = f(h_t^{'}, \bar{h})$. When then compute an attentional hidden state $\tilde{h}_t = \tanh(W_c[c_t; h_t^{'}])$, and calculate the predictive distribution over the next action as $a_t = \text{softmax}(\tilde{h}_t)$. Although visual attention has also proved highly beneficial in vision and language problems [60, 33, 3], we leave an investigation of visual attention in Vision-and-Language Navigation to future work.

5.2. Training

We investigate two training regimes, 'teacher-forcing' and 'student-forcing'. In both cases, we use cross entropy loss at each step to maximize the likelihood of the ground-truth target action a_t^* given the previous state-action sequence $\langle s_0, a_0, s_1, a_1, \ldots, s_t \rangle$. The target output action a_t^* is always defined as the next action in the ground-truth shortest-path trajectory from the agent's current pose $s_t = \langle v_t, \psi_t, \theta_t \rangle$ to the target location v^*.

Under the 'teacher-forcing' [32] approach, at each step during training the ground-truth target action a_t^* is selected, to be conditioned on for the prediction of later outputs. However, this limits exploration to only states that are in ground-truth shortest-path trajectory, resulting in a changing input distribution between training and testing [45, 32]. To address this limitation, we also investigate 'student-forcing'. In this approach, at each step the next action is sampled from the agent's output probability distribution. Student-forcing is equivalent to an online version of DAGGER [45], or the 'always sampling' approach in scheduled sampling [8][4].

Implementation Details We perform only minimal text pre-processing, converting all sentences to lower case, tokenizing on white space, and filtering words that do not occur at least five times. We set the simulator image resolution to 640×480 with a vertical field of view of 60 degrees. We set the number of hidden units in each LSTM to 512, the size of the input word embedding to 256, and the size of the input action embedding to 32. Embeddings are learned from random initialization. We use dropout of 0.5 on embeddings, CNN features and within the attention model.

As we have discretized the agent's heading and elevation changes in 30 degree increments, for fast training we extract and pre-cache all CNN feature vectors. We train in PyTorch using the Adam optimizer [28] with weight decay and a batch size of 100. In all cases we train for a fixed number of iterations. As the evaluation is single-shot, at test time we use greedy decoding [44]. Our test set submission is trained on all training and validation data.

---

[4]Scheduled sampling has been shown to improve performance on tasks for which it is difficult to exactly determine the best next target output a_t^* for an arbitrary preceding sequence (e.g. language generation [8]). However, in our task we can easily determine the shortest trajectory to the goal location from anywhere, and we found in initial experiments that scheduled sampling performed worse than student-forcing (i.e., always sampling).

	Trajectory Length (m)	Navigation Error (m)	Success (%)	Oracle Success (%)
Val Seen:				
SHORTEST	10.19	0.00	100	100
RANDOM	9.58	9.45	15.9	21.4
Teacher-forcing	10.95	8.01	27.1	36.7
Student-forcing	11.33	6.01	38.6	52.9
Val Unseen:				
SHORTEST	9.48	0.00	100	100
RANDOM	9.77	9.23	16.3	22.0
Teacher-forcing	10.67	8.61	19.6	29.1
Student-forcing	8.39	7.81	21.8	28.4
Test (unseen):				
SHORTEST	9.93	0.00	100	100
RANDOM	9.93	9.77	13.2	18.3
Human	11.90	1.61	86.4	90.2
Student-forcing	8.13	7.85	20.4	26.6

Table 1. Average R2R navigation results using evaluation metrics defined in Section 4.4. Our seq-2-seq model trained with student-forcing achieves promising results in previously explored environments (Val Seen). Generalization to previously *unseen* environments (Val Unseen / Test) is far more challenging.

5.3. Additional Baselines

Learning free We report two learning-free baselines which we denote as RANDOM and SHORTEST. The RANDOM agent exploits the characteristics of the dataset by turning to a randomly selected heading, then completing a total of 5 successful `forward` actions (when no `forward` action is available the agent selects `right`). The SHORTEST agent always follows the shortest path to the goal.

Human We quantify human performance by collecting human-generated trajectories for one third of the test set (1,390 instructions) using AMT. The collection procedure is similar to the dataset collection procedure described in Section 4.2, with two major differences. First, workers are provided with navigation instructions. Second, the entire scene environment is freely navigable in first-person by clicking on nearby viewpoints. In effect, workers are provided with the same information received by an agent in the simulator. To ensure a high standard, we paid workers bonuses for stopping within 3m of the true goal location.

6. Results

As illustrated in Table 1, our exploitative RANDOM agent achieves an average success rate of 13.2% on the test set (which appears to be slightly more challenging than the validation sets). In comparison, AMT workers achieve 86.4% success on the test set, illustrating the high quality of the dataset instructions. Nevertheless, people are not infallible when it comes to navigation. For example, in the dataset we occasionally observe some confusion between right and
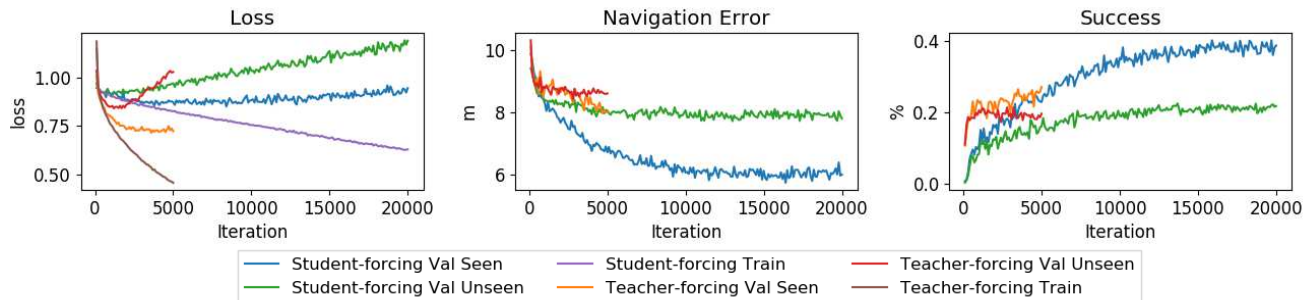
Figure 7. Validation loss, navigation error and success rate during training. Our experiments suggest that neural network approaches can strongly overfit to training environments, even with regularization. This makes generalizing to unseen environments challenging.
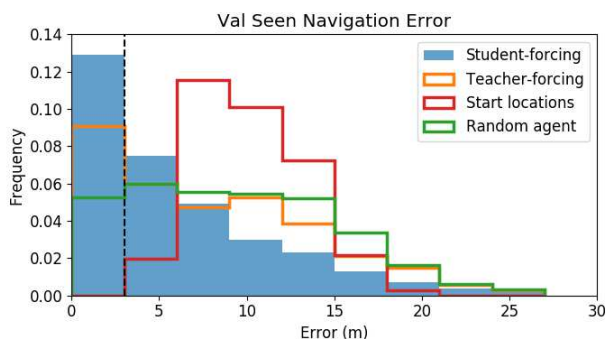

Figure 8. In previously seen environments student-forcing training achieves 38.6% success (< 3m navigation error).

left (although this is recoverable if the instructions contain enough visually-grounded references). In practice, people also use two additional mechanisms to reduce ambiguity that are not available here, namely gestures and dialog.

With regard to the sequence-to-sequence model, student-forcing is a more effective training regime than teacher-forcing, although it takes longer to train as it explores more of the environment. Both methods improve significantly over the RANDOM baseline, as illustrated in Figure 8. Using the student-forcing approach we establish the first test set leaderboard result achieving a 20.4% success rate.

The most surprising aspect of the results is the significant difference between performance in seen and unseen validation environments (38.6% vs. 21.8% success for student-forcing). To better explain these results, in Figure 7 we plot validation performance during training. Even using strong regularization (dropout and weight decay), performance in unseen environments plateaus quickly, but further training continues to improve performance in the training environments. This suggests that the visual groundings learned may be quite specific to the training environments.

Overall, the results illustrate the significant challenges involved in training agents that can generalize to perform well in previously unseen environments. The techniques

and practices used to optimize performance on existing vision and language datasets are unlikely to be sufficient for models that are expected to operate in new environments.

7. Conclusion and Future Work

Vision-and-Language Navigation (VLN) is important because it represents a significant step towards capabilities critical for practical robotics. To further the investigation of VLN, in this paper we introduced the Matterport3D Simulator. This simulator achieves a unique and desirable trade-off between reproducibility, interactivity, and visual realism. Leveraging these advantages, we collected the Room-to-Room (R2R) dataset. The R2R dataset is the first dataset to evaluate the capability to follow natural language navigation instructions in previously unseen real images at building scale. To explore this task we investigated several baselines and a sequence-to-sequence neural network agent.

From this work we reach three main conclusions. First, VLN is interesting because existing vision and language methods can be successfully applied. Second, the challenge of generalizing to previously *unseen* environments is significant. Third, crowd-sourced reconstructions of *real* locations are a highly-scalable and underutilized resource[5]. The process used to generate R2R is applicable to a host of related vision and language problems, particularly in robotics. We hope that this simulator will benefit the community by providing a visually-realistic framework to investigate VLN and related problems such as navigation instruction generation, embodied visual question answering, human-robot dialog, and domain transfer to real settings.

---

[5]The existing Matterport3D data release constitutes just 90 out of more than 700,000 building scans that have been already been collected [37].

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 4

[2] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, 2017. 3

[3] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018. 7

[4] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual question answering. In *ICCV*, 2015. 2, 4, 5

[5] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *arXiv preprint arXiv:1702.01105*, 2017. 4

[6] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 6

[7] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016. 1, 2, 3

[8] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015. 7

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016. 4

[10] S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville. HoME: A household multimodal environment. *arXiv:1711.11017*, 2017. 3

[11] 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. 1, 2, 3, 5, 6

[12] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017. 1, 2, 3

[13] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *AAAI*, 2011. 1, 2

[14] X. Chen, T.-Y. L. Hao Fang, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv preprint arXiv:1504.00325*, 2015. 2, 4

[15] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 3

[16] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *CVPR*, 2018. 2, 4

[17] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. F. Moura, D. Parikh, and D. Batra. Visual dialog. In *CVPR*, 2017. 2

[18] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. IQA: Visual question answering in interactive environments. In *CVPR*, 2018. 2, 4

[19] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *CVPR*, 2017. 2

[20] 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. 1, 2

[21] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017. 3

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[23] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997. 6

[24] 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. 2

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 4

[26] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. Referit game: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 2

[27] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, 2016. 1, 2, 3

[28] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[29] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 259–266. IEEE, 2010. 2

[30] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An interactive 3d environment for visual AI. *arXiv:1712.05474*, 2017. 3

[31] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016. 3

[32] A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*, 2016. 7

[33] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2016. 7

[34] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2014. 6

[35] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *AAAI*, 2006. 2

[36] J. Mao, H. Jonathan, A. Toshev, O. Camburu, A. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016. 2

[37] Matterport. Press release, October 2017. 8

[38] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, 2016. 1, 2

[39] A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017. 4

[40] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. In *ICLR*, 2017. 6

[41] D. K. Misra, J. Langford, and Y. Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP*, 2017. 1, 3, 6

[42] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 3

[43] D. A. Norman. *The Design of Everyday Things*. Basic Books, Inc., New York, NY, USA, 2002. 5

[44] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *CVPR*, 2017. 7

[45] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011. 7

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 6

[47] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 3, 4

[48] S. Song, S. P. Lichtenberg, and J. Xiao. SUN RGB-D: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 3

[49] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 6

[50] L. Tai and M. Liu. Towards cognitive exploration through deep reinforcement learning for mobile robots. *arXiv preprint arXiv:1610.01733*, 2016. 3

[51] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler. MovieQA: Understanding stories in movies through question-answering. In *CVPR*, 2016. 2

[52] 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. 1, 2

[53] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, pages 1553–1561, 2017. 3

[54] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 4

[55] A. Vogel and D. Jurafsky. Learning to follow navigational directions. In *ACL*, 2010. 2

[56] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, 2007. 6

[57] T. Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, Massachusetts Institute of Technology, 1971. 2

[58] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv:1801.02209*, 2018. 3

[59] C. Yan, D. Misra, A. Bennnett, A. Walsman, Y. Bisk, and Y. Artzi. CHALET: Cornell house agent learning environment. *arXiv:1801.07357*, 2018. 3

[60] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. 7

[61] A. R. Zamir, F. Xia, J. He, S. Sax, J. Malik, and S. Savarese. Gibson Env: Real-world perception for embodied agents. In *CVPR*, 2018. 3

[62] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 1, 2, 3, 4, 6