Waypoint Models for Instruction-guided Navigation in Continuous Environments

Jacob Krantz[1]* Aaron Gokaslan[2,3] Dhruv Batra[2,4] Stefan Lee[1] Oleksandr Maksymets[2]
[1]Oregon State University [2]Facebook AI Research [3]Cornell University
[4]Georgia Institute of Technology
Project Webpage: https://jacobkrantz.github.io/waypoint-vlnce

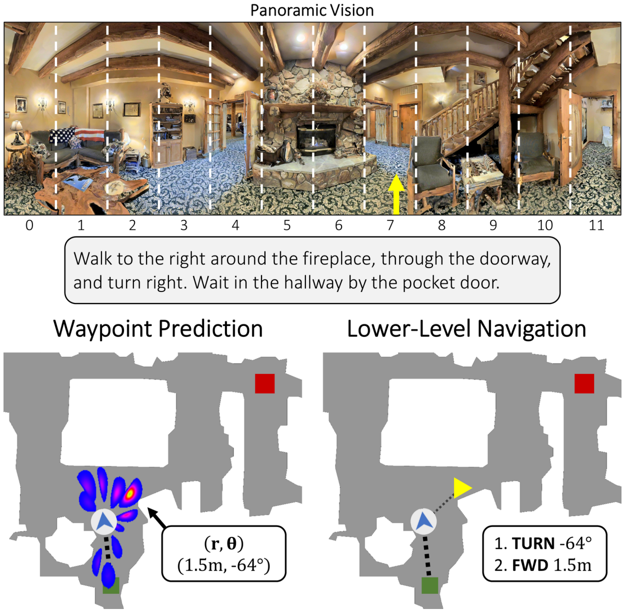
Abstract

Little inquiry has explicitly addressed the role of action spaces in language-guided visual navigation – either in terms of its effect on navigation success or the efficiency with which a robotic agent could execute the resulting trajectory. Building on the recently released VLN-CE [24] setting for instruction following in continuous environments, we develop a class of language-conditioned waypoint prediction networks to examine this question. We vary the expressivity of these models to explore a spectrum between low-level actions and continuous waypoint prediction. We measure task performance and estimated execution time on a profiled LoCoBot [1] robot. We find more expressive models result in simpler, faster to execute trajectories, but lower-level actions can achieve better navigation metrics by approximating shortest paths better. Further, our models outperform prior work in VLN-CE and set a new state-of-the-art on the public leaderboard – increasing success rate by 4% with our best model on this challenging task.

1. Introduction

A long-term goal of instruction-guided visual navigation research is to develop AI for robotic agents that can reliably follow paths described by natural language navigation instructions in new environments. Much of the existing work in this domain is robot-agnostic and has focused on highly-abstract simulators where agents navigate by choosing between a small, fixed set of nearby locations that the agent then transitions to deterministically [4, 15, 22, 25] – essentially assuming some underlying robot-specific control system can perform navigation. The Vision-and-Language Navigation (VLN) [4] task is representative of this class of problem settings.

In sim-to-real experiments, Anderson et al. [3] demonstrate that a major performance bottleneck for transferring VLN agents trained in high-level simulators to real robotic systems is producing appropriate sets of nearby locations (or waypoints) to choose from; however, it is infeasible to study waypoint prediction in the discrete, highly-abstract simulator as the agent can only occupy predefined locations.



Walk to the right around the fireplace, through the doorway, and turn right. Wait in the hallway by the pocket door.

Figure 1. Our approach decomposes the task of following navigation instructions in continuous environments into language-conditioned waypoint prediction and low-level navigation.

Recently, Krantz et al. [24] introduced a variant of VLN instantiated in continuous simulated environments (denoted VLN-CE) such that agents can move to arbitrary positions. In contrast to the highly-abstract action space in VLN, agents in [24] navigate by executing a sequence of low-level actions such as moving forward 0.25 meters or turning by 15 degree increments. This end-to-end, instruction-to-low-level-control design choice has implications both in simulation and for potential sim-to-real transfer to a robotic platform. During training, these policies must jointly learn

*Work done during an internship at Facebook AI Research. Correspondence: krantzja@oregonstate.edu navigation and language grounding over long sequences of actions ($\sim$55 per episode). As a result, [24] shows that models mirroring successful VLN agents perform substantially worse in VLN-CE.

On a real robot, the frequent stop, starts, and turns induced by this action space can be slow to execute (requiring frequent changes in velocity and calls to a planner), result in state estimation error, and strain hardware [23, 19]. Further, executing the deep policy network to predict actions at each time step can put extra demand on robot power supplies.

This work explores a spectrum of action spaces between these two extremes – studying instruction-guided navigators that predict relative waypoints with varied expressivity. At one end, our agents are free to predict relative waypoints as continuous points within some maximum range. On the other, the action space is reduced to taking a fixed step in a direction chosen from a small, finite set of angles – mimicking [24] but collapsing consecutive turns. In between, we experiment with mixing discrete and continuous components to parameterize waypoint predictions.

To do this, we develop an attention-based waypoint prediction network for instruction following. Given a navigation instruction and a panoramic RGBD observation at the current position, our agents predict a distribution over relative waypoints in polar coordinates (consisting of a heading angle θ and a distance r). A low-level continuous navigator is then executed to move in a straight line towards the waypoint – leaving concerns about obstacle avoidance to the waypoint predictor. We train our agents as model-free control policies using large-scale reinforcement learning [37] on the VLN-CE dataset. We evaluate our agents using standard metrics for VLN-CE as well as the estimated execution time for resulting trajectories on a LoCoBot [1] robot.

We find that more expressive waypoint prediction networks result in simpler paths that are faster to execute; however, more constrained action spaces can achieve better performance by more closely approximating shortest paths. Our waypoint models paired with continuous low-level navigators reduce the average estimated time to execute a trajectory by 2.2 times compared to low-level turn/forward models. When paired with discrete low-level navigators to match VLN-CE's action space, our models improve navigation success rate by 1-4% over prior work on the VLN-CE leaderboard[1] – a max relative improvement of 14%.

Contributions. We summarize our contributions as:

- Developing a class of language-conditioned waypoint prediction networks for the VLN-CE task,
- Providing empirical analysis of waypoint prediction expressivity's effect on navigation success and estimated time to execute trajectories on a representative robot,
- Demonstrating that our models paired with low-level navigators set a new state-of-the-art on the VLN-CE test leaderboard by an absolute 4% success rate.

We provide open-source code and pre-trained models at https://github.com/jacobkrantz/VLN-CE.

2. Related Work

Instruction-Guided Navigation. Many works have examined instruction-guided navigation. Task descriptions vary across a number of axes, including instruction source (templated [22], natural language [4]), instruction language (monolingual, multilingual [25]), environment setting (indoor, outdoor [15, 22]), environment realism (synthetic simulation [28], realistic simulation [4], real-world [8, 3]), navigation affordance (sparse navigation graphs [4], continuous space [24, 8]), and agent (ground-based, quadcopters [6]).

One popular task is Vision-and-Language Navigation (VLN) [4]. VLN has natural language instructions and uses indoor, photo-realistic environments from the Matterport3D dataset [10]. A ground-based agent acts on a sparse navigation-graph. In this work, we consider the recently released Vision-and-Language Navigation in Continuous Environments (VLN-CE) [24], a task that lifts VLN to continuous 3D environments. We explore waypoint models that leverage more abstract action spaces in VLN-CE.

Hierarchical Visual Navigation. Waypoint-based models can be considered a type of hierarchical agent, which has been proposed for many tasks relating to visual navigation. Beyond an intuitive problem decomposition, these are commonly motivated by a desire to carve out self-contained sub-tasks solvable with existing approaches [5], circumvent challenges faced by reinforcement learning (RL) algorithms (*e.g.* credit assignment and exploration over long time horizons) [12, 38], or to introduce interpretable representations [16]. However, these works address embodied navigation tasks that do not condition on language.

More related to our work are those that predict waypoints directly [5, 13, 14]. Chaplot *et al.* [13] address the image-goal navigation task with a topological agent that updates a graph with candidate "ghost nodes", selects a node to navigate to, and performs low-level navigation. Our waypoint-based model differs in that we combine the waypoint prediction and selection steps and condition both with the task goal. Chen *et al.* [14] take a similar approach to ours for audio-visual navigation – predicting waypoints conditioned on audio goals (e.g. a teapot whistling) while building a metric map. Our approach predicts waypoints directly from language instructions without a metric map.

Several hierarchical models have been developed for instruction-guided navigation tasks. For an outdoor environment, Misra *et al.* [28] decompose the task into goal prediction and action generation. While effective in (nearly) fully-observable environments, this method does not readily transfer to novel environments with partial observability. Likewise, Blukis *et al.* [7] develop a network that predicts

[1] eval.ai/web/challenges/challenge-page/719

and updates a position-visitation distribution en route to the goal. This approach leverages assumptions of an aerial vehicle operating in outdoor environments, namely, nearly full observability compared to an indoor ground-based agent and rare obstacle collisions afforded by aerial free-space.

Recent sim2real transfer work in VLN has considered adding a software harness that emulates an ‘online’ navigation graph by predicting candidate waypoints [3]. This mechanism is not conditioned on instructions and just uses local visual / lidar observations. VLN agents trained in topological simulators can then navigate on this graph by invoking a classical navigation stack in the real world. However, these models were found to perform significantly worse than when given a known navigation graph – suggesting that waypoint prediction remains a bottleneck for VLN transfer. Instead of a two-stage process, we present an alternative – developing a language-conditioned waypoint prediction network in a continuous simulator.

Training Instruction Followers. Many instruction-guided navigation works learn policies via imitation learning [4, 18, 6, 36, 34, 24, 26]. Behavior cloning can result in exposure bias. Methods like student forcing and dataset aggregation reduce this but require a queriable expert policy and discourage exploration [4, 32]. Some works train agents with a combination of imitation learning and reinforcement learning (RL) [8, 25]. In this work, we learn linguistically-motivated waypoint predictions purely from RL.

3. Task Description

We consider the episodic task of instruction-guided visual navigation in previously-unseen environments. An agent must navigate a path specified by natural language instructions and stop at a goal location. The agent has egocentric RGBD perception. The environment is continuous, requiring the agent to navigate freely about the 3D space and contend with obstacles and occlusion.

VLN-CE Task. We set our work in the context of the Vision-and-Language Navigation in Continuous Environments (VLN-CE) task [24]. VLN-CE is based on the Room-to-Room dataset used in the original VLN task [4]. VLN has agents navigate on a pre-defined graph of viewpoints with scenes from the Matterport3D dataset [10]. VLN-CE replaces the viewpoint topology with full Matterport3D scene reconstructions, lifting the VLN task to more realistic navigation in continuous space. We conduct our experiments in VLN-CE because it enables our study of agents that predict arbitrary relative waypoints. We adopt the task settings of VLN-CE with specific extensions detailed below.

Observation Space. The agent observes RGB and depth images. For both modalities, we extend the 90° HFOV of VLN-CE to panoramic 360° HFOV. Each panorama is captured as twelve frames angled in 30° increments, where each frame has a 90° HFOV at a resolution of 256 x 256. Panoramic vision is common in related visual navigation tasks like VLN and PointGoal navigation [18, 13] and panoramic sensors could be used in real applications.

Action Space. Waypoint-based agents can operate independently of the low-level action space used to reach the predicted waypoints. We experiment with two action spaces that operate in discrete time. Specifically, we train and evaluate our agent with continuous-space actions that specify real-valued turn angles and straight-line distances. Such actions can be accomplished by zero-turn-radius robots such as Locobot [20]. Like VLN-CE, we assume perfect actuation to keep results comparable. We also evaluate our agent with the VLN-CE's discrete action space to enable direct comparison with past work (`forward` 0.25m, `left` 15 degrees, `right` 15 degrees, and `stop`). Actions specifying velocities or accelerations are beyond the scope of this work, but are compatible with waypoint-based agents [5].

4. Method

We describe our implementation of a waypoint-based instruction-following agent. A waypoint prediction network (WPN) predicts navigation waypoints or a `STOP` action directly from pixels and natural language. Waypoints are passed to a lower-level navigator in relative polar coordinates. We employ a simple two-step navigator that turns in the direction of the waypoint then moves forward the predicted distance. This navigator does no direct language processing, separating the task between two agent components.

4.1. Waypoint Prediction Network (WPN)

An overview of our network is shown in Fig. 2. At each time step, our agent observes the world through a panoramic RGBD sensor represented by 12 RGBD observations captured at regular angular intervals (θ = 0, 15, 30, ..., 330). Our agent predicts the next navigation waypoint in relative coordinates by selecting one of these discrete observations as a coarse heading $\hat{\theta}^D$ and then predicting an angular offset $\hat{\theta}^{offset}$ and distance $\hat{r}$ such that the waypoint is specified by the polar coordinates $(\hat{r}, \hat{\theta}^D + \hat{\theta}^{offset})$. We base our model architecture on the cross-modal attention network of Krantz *et al.* [24], adapting the single-modality visual encoders to panoramas, adding attention over panorama frames, and developing action generation layers for waypoint prediction.

Visual Encoding. Our network encodes RGB and depth observations separately. Each RGB frame is encoded with a ResNet-18 [21] pre-trained on ImageNet, collectively producing features $\mathcal{V}_t \in \mathbb{R}^{12 \times i \times j}$ for 12 frames containing i feature map channels of flattened spatial dimensions j. Similarly, each depth frame is encoded with a ResNet-50 pre-trained on a PointGoal navigation task [37], collectively producing features $\mathcal{D}_t \in \mathbb{R}^{12 \times k \times l}$. We provide static pose features $\mathcal{P} \in \mathbb{R}^{12 \times 2}$ consisting of the sine and cosine of the camera angle. These features disambiguate the rela-

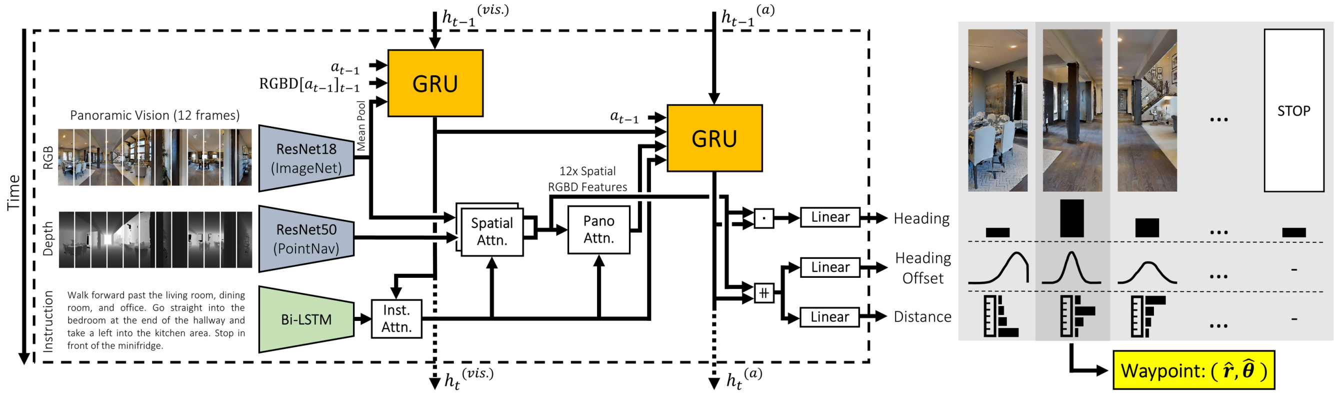

$h_{t-1}^{(vis.)}$
$h_{t-1}^{(a)}$
a_{t-1}
$\mathrm{RGBD}[a_{t-1}]_{t-1}$
GRU
Mean Pool
a_{t-1}
GRU
Time
Panoramic Vision (12 frames)
RGB
Depth
Instruction
ResNet18 (ImageNet)
ResNet50 (PointNav)
Bi-LSTM
Walk forward past the living room, dining room, and office. Go straight into the bedroom at the end of the hallway and take a left into the kitchen area. Stop in front of the minifridge.
Inst. Attn.
Spatial Attn.
Pano Attn.
12x Spatial RGBD Features
·
‖
Linear
Linear
Linear
Heading
Heading Offset
Distance
STOP
...
Waypoint: ($\hat{\boldsymbol{r}}, \hat{\boldsymbol{\theta}}$)
$h_t^{(vis.)}$
$h_t^{(a)}$

Figure 2. We develop a waypoint prediction network (WPN) that predicts relative waypoints directly from natural language instructions and panoramic vision. Our WPN uses two levels of cross-modal attention and prediction refinement to align visual observations with actions.

tive angle between frames and are commonly used by VLN panorama agents to encode previous actions [18].

The difference between an agent's visual observations at time t *vs.* $t-1$ can be more substantial with waypoint-based navigation than with a lower-level action space, *e.g.*, when a waypoint is predicted through a doorway. We provide this visual context explicitly by including a subset of visual features from the previous time step. Specifically, we include features for the panorama frame facing nearest the heading of the last waypoint prediction: $\mathcal{V}_{t-1}^{(i)}$ and $\mathcal{D}_{t-1}^{(i)}$ where $i = \hat{\theta}_{t-1}^{D}$. These features are mean-pooled across their spatial dimension, resulting in a visual context vector $\bar{\mathcal{C}} = [\bar{\mathcal{V}}_{t-1}^{(i)}, \bar{\mathcal{D}}_{t-1}^{(i)}]$, where $[\cdot]$ denotes concatenation.

Instruction Encoding. We use the same instruction encoding as Krantz *et al.* [24]. The natural language instruction $\mathcal{O}^{\text{inst.}}$ is a lightly-tokenized sequence of words observed at each time step. We map $\mathcal{O}^{\text{inst.}}$ to a sequence of GloVE [31] embeddings $w_1, w_2, ..., w_{\text{N}}$ for an instruction of length N words. A bi-directional LSTM then produces hidden states

$$\mathcal{S} = \{s_1, s_2, ..., s_{\text{N}}\} = \text{BiLSTM}(w_1, w_2, ..., w_{\text{N}}). \quad (1)$$

Previous Action Encoding. Our network observes the predicted waypoint from the previous time step as a vector $a_{t-1} = [\hat{r}_{t-1}, \sin(\hat{\theta}_{t-1}^{D}), \cos(\hat{\theta}_{t-1}^{D}), \hat{\theta}_{t-1}^{offset}]$.

Visual History. We use a dedicated recurrent network to track visual history like Krantz *et al.* [24], including inputs of RGB features $\mathcal{V}_t$, the previous action a_{t-1}, and the additional visual context $\bar{\mathcal{C}}$. We mean-pool $\mathcal{V}_t$ across both the spatial and frame dimensions, resulting in vector $\bar{\mathcal{V}}_t$. Our visual history is then encoded as

$$h_t^{(vis.)} = \text{GRU}\left(\left[\bar{\mathcal{V}}_t, \bar{\mathcal{C}}, a_{t-1}\right], h_{t-1}^{(vis.)}\right). \quad (2)$$

Cross-Modal Attention. We use scaled dot-product attention (Attn) for all attention mechanisms in our network [35]. The output of the visual history module $h_t^{(vis.)}$ attends to the recurrent instruction features $\mathcal{S}$:

$$\hat{\mathcal{S}} = \text{Attn}\left(\mathcal{S}, h_t^{(vis.)}\right). \quad (3)$$

These attended instruction features are then used to perform spatial attention on each RGB and depth frame i:

$$\hat{\mathcal{V}}_t^{(i)} = \text{Attn}\left(\bar{\mathcal{V}}_t^{(i)}, \hat{\mathcal{S}}\right) \qquad \hat{\mathcal{D}}_t^{(i)} = \text{Attn}\left(\bar{\mathcal{D}}_t^{(i)}, \hat{\mathcal{S}}\right) \quad (4)$$

which is shown in Fig. 2 as `Spatial Attn`. The resulting features are concatenated with pose features $\mathcal{P}$, resulting in instruction-conditioned and heading-aware RGBD features for each panorama frame:

$$\hat{\mathcal{I}}_t = \left[\hat{\mathcal{V}}_t, \hat{\mathcal{D}}_t, \mathcal{P}\right]. \quad (5)$$

The attended instruction features are used again to attend across panorama frames (`Pano Attn` in Fig. 2) prior to a final recurrent block:

$$\hat{\mathcal{X}} = \text{Attn}\left(\hat{\mathcal{I}}_t, \hat{\mathcal{S}}\right) \quad (6)$$

$$h_t^{(a)} = \text{GRU}\left(\left[\hat{\mathcal{X}}, \hat{\mathcal{S}}, h_t^{(vis.)}, a_{t-1}\right], h_{t-1}^{(a)}\right). \quad (7)$$

Action Prediction. We use the final recurrent state $h_t^{(a)}$ and the frame-specific features $\hat{\mathcal{I}}_t$ to predict a waypoint in relative polar coordinates. Our waypoint prediction begins as a coarse heading prediction sampled from a distribution over the 12 frames and a `STOP` action: $\hat{\theta}^D \sim$ `Pano`. The logits of `Pano` are the dot product between $h_t^{(a)}$ and $\hat{\mathcal{I}}_t$:

$$\texttt{Pano} = \text{softmax}\left(\left[\hat{\mathcal{I}}_t \cdot h_t^{(a)}, W_s h_t^{(a)} + b_s\right]\right). \quad (8)$$

For each frame heading i in `Pano`, we predict distributions over a heading offset refinement and a distance as shown in Fig. 2. In Sec. 5.2, we explore how the expressivity of the waypoint action space affects performance. To support those experiments, our offset and distance distributions are either continuous, discrete, or constant. We use the truncated Gaussian distribution [9] for fixed-range continuous predictions and parameterize it by predicting the mean and variance of the underlying Gaussian:

$$\texttt{Offset}^{(i)} = \tanh\left(W_o\left[\hat{\mathcal{I}}_t^{(i)}, h_t^{(a)}\right] + b_o\right) \text{ and} \quad (9)$$

$$\texttt{Dist}^{(i)} = \text{sigmoid}\left(W_d\left[\hat{\mathcal{I}}_t^{(i)}, h_t^{(a)}\right] + b_d\right) \quad (10)$$

where the range of $\texttt{Offset}^{(i)}$ is $[-15°, 15°]$ and the range of $\texttt{Dist}^{(i)}$ is $[0.25\text{m}, 4.0\text{m}]$. For discrete distributions, we replace the tanh and sigmoid activation functions with a softmax for an offset domain of $\{-15°, -10°, , ..., 15°\}$ and a distance domain of $\{0.25\text{m}, 0.75\text{m}, ..., 2.75\text{m}\}$. For constant predictions, the offset is $0°$ and the distance is 0.25m, corresponding to the forward step size of the standard VLN-CE action space.

We sample a heading offset $\hat{\theta}^{offset} \sim \texttt{Offset}^{(\hat{\theta}^D)}$ and a distance $\hat{r} \sim \texttt{Dist}^{(\hat{\theta}^D)}$ conditioned on the chosen coarse heading $\hat{\theta}^D$. This produces a polar waypoint prediction $(\hat{r}, \hat{\theta}^D + \hat{\theta}^{offset})$. We visualize a set of possible waypoint action spaces in Tab. 1.

4.2. Training the Waypoint Prediction Network

Existing work on the VLN-CE task trains agents with imitation learning [24]. Motivated by recent advancements in embodied navigation, we instead train our waypoint prediction network with decentralized distributed proximal policy optimization (DDPPO) [37]. DDPPO is a scaled version of the proximal policy optimization (PPO) algorithm with an actor-critic policy structure [33]. We consider the loss function used in [37] for PointGoal navigation. It employs the clipped PPO objective L_{action}, a clipped critic loss L_{value}, and an entropy bonus L_S to encourage exploration:

$$L_{\text{standard}} = L_{\text{action}} + c_v L_{\text{value}} - c_e L_S. \quad (11)$$

Let θ-parameterized policy π_θ be the waypoint prediction network. For L_{action}, we compute the probability $\pi_\theta(\mathcal{A}_t)$ of an action $\mathcal{A}_t = (\hat{\theta}^D, \hat{\theta}^{offset}, \hat{r})$ for a panorama frame selection $\hat{\theta}^D$, a heading offset $\hat{\theta}^{offset}$, and a distance $\hat{r}$ as:

$$\texttt{Pano}(\hat{\theta}^D) * \texttt{Offset}^{(\hat{\theta}^D)}(\hat{\theta}^{offset}) * \texttt{Dist}^{(\hat{\theta}^D)}(\hat{r}). \quad (12)$$

Accordingly, we define the entropy term L_S as:

$$L_S = c_p S\left(\texttt{Pano}\right) + c_o S\left(\texttt{Offset}\right) + c_d S\left(\texttt{Dist}\right) \quad (13)$$

to control the amount of exploration within specific action components. For L_{value}, we predict a state-value estimate from the final hidden state $h_t^{(a)}$ as $\hat{v} = \text{linear}(h_t^{(a)})$.

We expand this loss function with an additional zero-trending regularization term $L_{\text{offset}} = \left|\hat{\theta}^{offset}\right|$, which we found empirically led to better exploration of the joint $\texttt{Pano-Offset}$ heading space. Together, this yields our total loss function:

$$L_{\text{total}} = L_{\text{standard}} + c_r L_{\text{offset}}. \quad (14)$$

Reward Function. Our reward function is informed by the extrinsic reward structure of Wang *et al.* [36] and the time penalty (or slack reward) from Savva *et al.* [27]. We include a success reward r_{success}, the change in distance to target $\Delta_{\text{dist\_to\_target}}$ and a slack reward r_{slack}:

$$r(s,t) = r_{\text{success}} - \Delta_{\text{dist\_to\_target}} + r_{\text{slack}}, \quad (15)$$

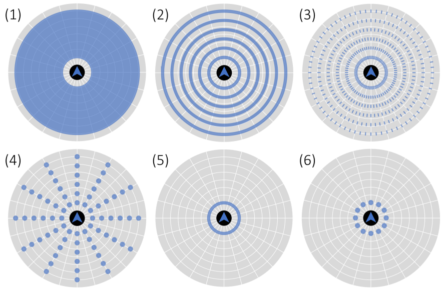
where $r_{\text{success}} = 2.5$ once a stop action is called within 3m of the target location (otherwise equal to 0) and $\Delta_{\text{dist\_to\_target}} = D(s_t) - D(s_{t-1})$ is progress towards the goal in terms of geodesic distance. The slack reward as defined by [27] is constant and applied at every time step. A waypoint predictor that maximizes this reward term would predict the furthest navigable waypoint toward the goal. This is undesirable for instruction-following where agents need to consider intermediate navigation decisions in light of partial observability. In the instruction *"go into the bedroom"*, an agent must first decide to continue past other similar-looking doorways (such as a bathroom) before choosing to enter the bedroom. We mitigate this training bias toward distant waypoints by scaling the slack reward based on waypoint distance instead of time. Specifically, we scale slack based on distance predicted: $r_{\text{slack}} = -0.05 \cdot \frac{d_{predicted}}{0.25m}$ which additionally penalizes unreachable waypoints.

5. Experiments

In this section, our main experiments address the following questions within the context of the VLN-CE task:

1) How does the expressivity of waypoint predictions affect performance? On one end of the expressivity spectrum, an agent may select waypoints from a small set of discrete candidates, and on the other end, an agent may consider any continuous location within some range. We examine the impact of different levels of expressivity in Sec. 5.2. Generally, we find that less expressive action spaces lead to minor improvements in standard metrics over more expressive versions but result in trajectories that would be slower to execute on real agents due to frequent stops and turns.

2) How do our waypoint-based models compare to prior work in low-level action spaces? Compared to existing work on VLN-CE [24], our base models are trained with additional sensors (forward-facing *vs.* panoramic cameras) and continuous navigators with arbitrary turn angles and step distances. While we argue these observation and navigator action spaces are *more reflective* of real robotic agents, we ablate these in Sec. 5.3 to compare with prior work. We find that our models result in significant improvements over prior work on the public VLN-CE leaderboard.



#	Model	Dist.	Offset	Val-Seen					Val-Unseen						
				TL	NE ↓	OS ↑	SR ↑	SPL ↑	TL	NE ↓	OS ↑	SR ↑	SPL ↑	EET	SCT ↑
1	Waypoint Pred. Network (WPN)	C	C	10.29	6.05	51	40	35	10.38	6.90	41	34	29	186	20
2		D	C	10.51	6.12	49	38	33	10.62	6.62	**43**	36	30	153	23
3		D	D	9.11	6.57	44	35	32	8.23	7.48	35	28	26	93	20
4		D	-	9.06	6.45	46	39	35	8.16	7.20	38	31	28	90	22
5	Heading Pred. Network (HPN)	-	C	8.71	**5.17**	53	**47**	**45**	7.71	**6.02**	42	38	**36**	297	11
6		-	-	8.63	5.44	51	44	42	7.72	6.21	38	34	32	308	11

Table 1. Results of our waypoint model on Val-Seen and Val-Unseen splits using the continuous navigator to reach waypoints. We demonstrate the influence of our action space components by successively constraining the waypoint action space. We find that the least-constrained heading prediction network performs the best according to conventional VLN metrics across both validation splits.

5.1. Experimental Setup

VLN-CE Dataset. We use the VLN-CE dataset [24] which consists of $16,844$ path-instruction pairs (5,611 unique paths) across 90 scenes. The dataset is split into train (Train), seen validation (Val-Seen), unseen validation (Val-Unseen), and test (Test). Both Val-Unseen and Test contain scenes the agent has not been exposed to during training.

Metrics. We evaluate our agent using established metrics from VLN-CE [24]. Specifically, we report the metrics used by the VLN-CE Challenge leaderboard which include trajectory length (TL), navigation error (NE), oracle success rate (OS), success rate (SR), and success weighted by inverse path length (SPL). Note that success occurs when an agent invokes the stop action within 3m of the goal. Please see [2, 4] for a detailed description of these metrics.

Implementation Details. We implement our agents in PyTorch [30] and use the Habitat Simulator [27]. We extend Habitat's DDPPO [37] training implementation to the VLN-CE task and add components for training waypoint prediction agents. We distribute training across 64 GPUs, collecting around 200M steps of experience to reach peak performance (5 days on average). We use the same set of hyperparameters for each experiment and include those values in the supplementary. We use early stopping during the training process and select the checkpoint with the highest SPL on Val-Unseen for all models. During the evaluation, the waypoint prediction network takes the mode of each action distribution which leads to deterministic results.

5.2. Impact of Waypoint Expressivity

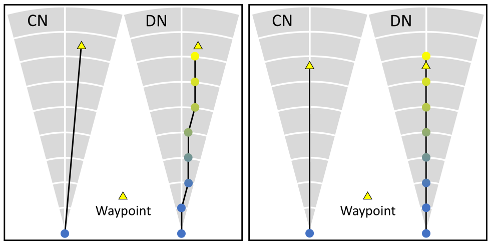
To study the effect of waypoint expressivity, we consider a spectrum of prediction domains for our model's distance and offset components. In Tab. 1, we consider predicting continuous values (C), choosing between a set of discrete values (D), or not predicting at all and using a fixed value (-). These combinations result in decision spaces visualized in the figure on the left of Tab. 1 where blue-shaded regions reflect possible waypoints under various C/D/- settings of offset and distance prediction. The labels at the top-left of each graph match the corresponding row(s) of the table.

WPN. Row 1 is our fully continuous waypoint prediction network (WPN) which can select any point within a toroid around the agent bounded by 0.25 and 4m. In row 2, we consider discrete distance prediction over six choices ranging from 0.25m to 2.75m in increments of 0.5m – resulting in a decision space of six continuous rings. In row 3, we additionally constrain the offset to seven choices ranging from $-15°$ to $15°$ in increments of $5°$ – further segmenting the rings into a dense set of discrete points. In row 4, we fix the offset prediction to 0, resulting in a sparse 'wagon-wheel' of 36 possible waypoints. This progresses from fully continuous to highly constrained subspaces.

We observe the most significant differences in performance from changes to the offset prediction space. Continuous offsets outperform their discrete or fixed counterparts by 3-8% success (rows 1 & 2 *vs.* 3 & 4). Intuitively, continuous offset prediction enables more position control at longer distances (compare the outer edge of plot 1 with 4). Surprisingly the dense discrete setting (row 3) under-performed no offsets (row 3 *vs.* 4) by 3% success. We suspect this is due to differences in training dynamics – we observe rapid training convergence for this model which could lead to relative under-exploration of the action space.

HPN. In rows 5 & 6 we ablate distance prediction entirely, moving a fixed 0.25m in the chosen heading. To reflect this, we refer to these ablations as Heading Prediction Networks (HPNs). For a continuous offset (row 5), this allows a waypoint to be predicted in a single ring of radius 0.25m. Row 6 further ablates the offset prediction, resulting in a "pick-pano" model that effectively mimics the existing VLN-CE action space Forward-Left-Right with collapsed turn actions (e.g. reducing any consecutive sequence of turns followed by a forward step into a single action). As before, we observe continuous offsets lead to improvements.

Counter-intuitively, we find these fixed-distance models generally outperform their WPN counterparts in terms of success by 2-3% (e.g. rows 1/2 *vs.* 5 and row 3 *vs.* 6).



#	Model	Navigator	Ckpt	Val-Seen					Val-Unseen				
				TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
1	WPN, discrete distance	CN	222	10.51	6.12	49	38	33	10.62	6.62	43	36	30
2		DN	222	9.64	6.33	43	34	30	9.54	6.85	40	33	28
3		DN	89	9.52	6.23	45	37	33	9.86	6.93	40	33	29
4	WPN, continuous distance	CN	137	10.29	6.05	51	40	35	10.38	6.90	41	34	29
5		DN	137	10.14	5.99	52	42	36	9.60	6.87	39	32	28
6		DN	185	10.73	5.99	52	41	36	10.61	7.07	42	33	28

Table 2. Validation performance of our waypoint prediction network (WPN) paired with different navigators. Despite training with a continuous navigator (CN), our WPN drops only 1-2 SPL in Val-Unseen using a discrete navigator (DN).

#	Model	Val-Seen					Val-Unseen					Test				
		TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
1	HPN + DN (ours)	8.54	**5.48**	**53**	**46**	**43**	7.62	**6.31**	**40**	**36**	**34**	8.02	**6.65**	**37**	**32**	**30**
2	WPN + DN (ours)	9.52	6.23	45	37	33	9.86	6.93	**40**	33	29	9.68	7.49	36	29	25
3	CMA+PM+DA+Aug [24]	9.06	7.21	44	34	32	8.27	7.60	36	29	27	8.85	7.91	36	28	25

Table 3. Results on the VLN-CE Challenge leaderboard. Both of our model submissions outperform existing state of the art on Test, with our heading prediction network (HPN) showing the highest success rate (SR) with the lowest trajectory length (TL).

However, these agents take approximately 4x the actions per trajectory (row 2 at 8.41 and row 5 at 33.41)– resulting in paths with significantly more starts, stops, and turns. Consequently, these high action-rate paths more closely approximate the ground truth path and achieve higher path efficiency as shown in SPL. In contrast, the WPN models break the path down into straight-line segments of 1.6 meters on average – reducing time to execute on real systems.

Given the variance associated with RL training methods, we repeat the experiment in row 2 of Tab. 1 twice under different random seeds. Both achieve a 29 SPL in Val-Unseen (1 point lower than row 2), suggesting that performance differences of 1 SPL may not be significant.

Path Efficiency under a LoCoBot Motion Model. Depending on a robot's abilities, shorter-length paths with many fine-grained actions may take considerably longer to execute than simpler-but-longer ones. We profile a LoCoBot [20] robot controlled via PyRobot [29]. We choose LoCoBot because it is a common platform for sim2real experiments in embodied tasks [23, 17, 11]. We derive functions for the time to turn by a specified angle or move forward by a specified distance from empirical measurements. Using these, we can estimate the time a LoCoBot would take for any path. For more details, see the supplementary.

We call this metric the estimated execution time (EET) and present results for each model in unseen environments in Tab. 1. We report EET in seconds. Intuitively, we find that models that predict travel distance (rows 1-4) have a lower EET than models that step in fixed 0.25m increments (rows 5-6). In particular, our best WPN (row 2) has a lower EET than our best HPN (row 5) by 144 seconds—nearly a 2x reduction. Digging into this further, we can compare the estimated average speed during a trajectory by normalizing trajectory length by EET (TL/EET). Our best WPN averages 6.9 cm/s, a 2.7x increase over our best HPN at just 2.6 cm/s.

We additionally present success weighted by completion time (SCT) [39] which scales the agent's success by the relative time to complete the trajectory. We adapt SCT to our agent's dynamics by using EET for completion times. Details are in the supplementary. We find that our best WPN model has over a 2x improvement in SCT over our best HPN model (23 *vs*. 11) despite WPN having a lower SPL. These results demonstrate the practical benefit of using waypoint models for real-world execution.

5.3. Comparison with Discrete Action Models

Our agents are trained with continuous navigators that can turn to arbitrary angles and move forward by arbitrary distances – matching realistic zero-turn radius robots. In contrast, VLN-CE assumes turns of 15 degree increments and forward steps of 0.25m. To compare with prior work, we implement a discrete navigator (DN) that uses this low-level action space to reach waypoints approximately. Our DN assumes free space and selects actions that greedily minimize distance to the waypoint. We assume no explicit localization. Tab. 2 shows our WPN model using continuous *vs*. discrete navigators at inference. As shown in the figure (left), the discrete navigator approximates the path of the continuous version. We find our models are somewhat robust to this change in navigator but drop 1-3% success. In rows 3 & 6 we re-evaluate all model checkpoints using the discrete navigator, finding that while different checkpoints maximize SPL, the performance is similar to rows 2 & 5.

In Tab. 3, we compare our models using a discrete nav-

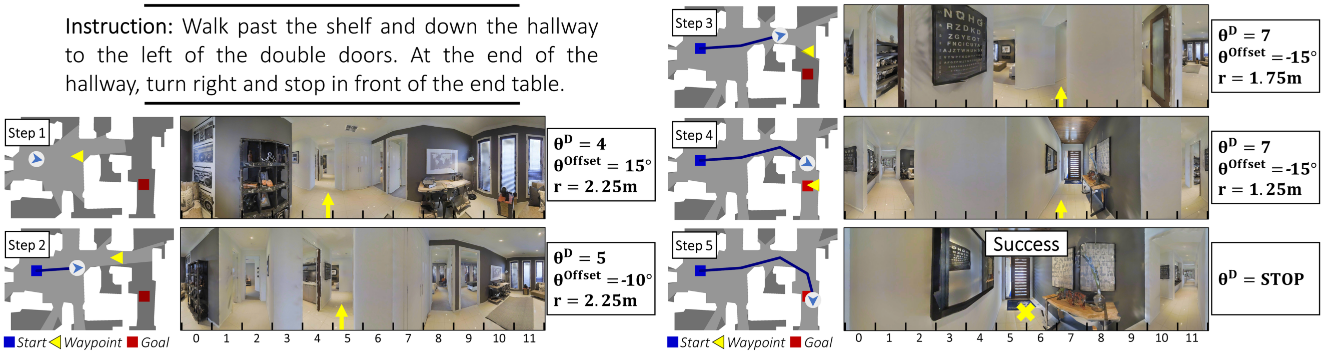


Figure 3. A qualitative example of our best waypoint agent (WPN+CN) successfully navigating to the goal in an unseen environment.

igator (DN) with prior work on the VLN-CE leaderboard. We submit both our best performing waypoint prediction network (WPN) and heading prediction network (HPN) variants based on Val-Unseen SPL. The existing state-of-the-art belongs to a cross-modal attention model trained by dataset aggregation (DAgger) and aided by progress monitor and data augmentation (CMA+PM+DA+Aug) [24]. Both WPN+DN and HPN+DN surpass the performance of existing work, with HPN+DN setting the new state of the art on the VLN-CE task by 4 SR (14% relative) and 5 SPL (20% relative). This is despite evaluating our networks with a navigator they were not exposed to during training.

Looking closely at the differences between prior work and our HPN+DN model, our agent has access to panoramic observations, has a more abstract heading-based action space, and is trained with RL. To ablate these differences, we start from our "pick-pano" HPN model (Tab. 1 row 6) and ablate panoramic observation to a single forward-facing camera. In Val-Unseen, this model surpasses row 6 by 2 SR (achieving a 36/32 SR/SPL). This agent has a lower seen-to-unseen gap than row 6 by 4 SR and 5 SPL (Val-Seen: 40/37 SR/SPL). This suggests that the reduced visual information of this model leads to less overfitting of the training environments. We further ablate the high-level action space of this model, matching the observation and action spaces of prior work. We find that this agent is unable to train to convergence after 300M steps of experience and identify the longer time horizon as a challenge requiring deeper study.

5.4. Qualitative Example

We present a qualitative example of our best waypoint agent navigating an unseen environment (Fig. 3). In Step 1, the agent traverses a large room by predicting a waypoint 2.25m away. In Step 4, the waypoint prediction is shorter at 1.25m, directly in front of the end table referenced in the instruction. Together, these predictions demonstrate the agent's ability to implicitly reason about scene geometry and predict language-grounded waypoints. Each step in this example can be aligned with an abstract semantic sub-goal, *e.g.* "*continue through the hallway*" (Step 2) and "*go to the end table*" (Step 4). Agents that directly predict actions from the VLN-CE action space need to make 10+ decisions to execute each sub-goal – an unintuitive and time-inefficient exercise. We provide additional navigation examples in the supplementary.

5.5. Waypoint Prediction Analysis

We analyze characteristics of the waypoints predicted by our best WPN model (Tab. 1 row 2). In both Val-Seen and Val-Unseen, the mean distance prediction is 1.6m with a standard deviation of 0.8m. We find that waypoint distances decrease with time, such that that predictions in the first 25% of an episode average 2.3m, predictions in the middle 50% average 1.6m, and predictions in the final 25% average 0.76m. This behavior is reasonable in the context of instruction-following – commonly, the beginning of a path is described as taking macro actions (*e.g.* "*Exit the bedroom...*"), while the end of a path can be described more particularly (*e.g.* "*...and wait between the two chairs.*").

6. Discussion

In this work, we present a model class that predicts navigation waypoints directly from language and vision. In exploring the expressivity of the waypoint action space, we find that more expressive models have favorable real-world execution properties, including a 2x reduction in expected execution time and a modular architecture that abstracts interaction with robot-specific navigation stacks. On the other hand, less expressive action spaces lead to higher traditional VLN metrics. Our best submission to the VLN-CE leaderboard demonstrates this through a 4% improvement in success (14% relative) and a 5 point improvement in SPL (20% relative) over prior work. We recognize that a significant gap still remains between topological VLN and continuous VLN-CE. Addressing this gap and the related sim2real gap [3] will require developing an effective interface between language understanding and robotic control.

Acknowledgements We would like to thank Naoki Yokoyama for helping adapt SCT and Joanne Truong for help with physical LoCoBot profiling. This work is funded in part by DARPA MCS.

References

[1] Locobot: an open source low cost robot. 2019. 1, 2

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 6

[3] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *CoRL*, 2020. 1, 2, 3, 8

[4] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1, 2, 3, 6

[5] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *CoRL*, pages 420–429, 2020. 2, 3

[6] Valts Blukis, Nataly Brukhim, Andrew Bennett, Ross A Knepper, and Yoav Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *RSS*, 2018. 2, 3

[7] Valts Blukis, Dipendra Misra, Ross A Knepper, and Yoav Artzi. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *CoRL*, 2018. 2

[8] Valts Blukis, Yannick Terme, Eyvind Niklasson, Ross A Knepper, and Yoav Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *CoRL*, pages 1415–1438, 2020. 2, 3

[9] John Burkardt. The truncated normal distribution. *Department of Scientific Computing Website, Florida State University*, pages 1–35, 2014. 5

[10] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: learning from rgb-d data in indoor environments. In *3DV*, 2017. MatterPort3D dataset license available at: `http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf`. 2, 3

[11] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 7

[12] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *NeurIPS*, 2020. 2

[13] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, pages 12875–12884, 2020. 2, 3

[14] Changan Chen, Sagnik Majumder, Al-Halah Ziad, Ruohan Gao, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Learning to set waypoints for audio-visual navigation. In *ICLR*, 2021. 2

[15] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*, 2019. 1, 2

[16] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. In *CoRL*, 2018. 2

[17] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: an open simulation-to-real embodied ai platform. In *CVPR*, pages 3164–3174, 2020. 7

[18] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018. 3, 4

[19] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE T-ITS*, 17(4):1135–1145, 2015. 2

[20] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: improving generalization and reducing dataset bias. *NeurIPS*, pages 9094–9104, 2018. 3, 7

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[22] Karl Moritz Hermann, Mateusz Malinowski, Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, and Raia Hadsell. Learning to follow directions in street view. *AAAI*, 2020. 1, 2

[23] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation. In *IROS*, 2020. 2, 7

[24] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *ECCV*, pages 104–120, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[25] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: multilingual vision-and-language navigation with dense spatiotemporal grounding. In *EMNLP*, pages 4392–4412, 2020. 1, 2, 3

[26] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *ECCV*, pages 259–274, 2020. 3

[27] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: a platform for embodied ai research. *ICCV*, 2019. 5, 6

[28] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In *EMNLP*, 2018. 2

[29] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav

Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019. 7

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6

[31] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: global vectors for word representation. In *EMNLP*, 2014. 4

[32] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011. 3

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5

[34] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL HLT*, 2019. 3

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[36] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019. 3, 5

[37] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2020. 2, 3, 5, 6

[38] Fei Xia, Chengshu Li, Roberto Martín-Martín, Or Litany, Alexander Toshev, and Silvio Savarese. Relmogen: leveraging motion generation in reinforcement learning for mobile manipulation. *arXiv preprint arXiv:2008.07792*, 2020. 2

[39] Naoki Yokoyama, Sehoon Ha, and Dhruv Batra. Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation. *arXiv preprint arXiv:2103.08022*, 2021. 7