Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments – Extended Abstract

Anonymous Authors[1]

Abstract

We develop a language-guided navigation task set in a continuous 3D environment where agents must execute low-level actions to follow natural language navigation directions. By being situated in continuous environments, this setting lifts a number of assumptions implicit in prior work that represents environments as a sparse graph of panoramas with edges corresponding to navigability. Specifically, our setting drops the presumptions of known environment topologies, short-range oracle navigation, and perfect agent localization. To contextualize this new task, we develop models that mirror many of the advances made in prior settings. We find significantly lower performance in the continuous setting – suggesting that performance in topological settings may be inflated by the strong implicit assumptions.

1. Introduction

Springing forth from the pages of science fiction and capturing the daydreams of weary chore-doers everywhere, the promise and potential of general-purpose robotic assistants that follow natural language instructions has been long understood. Taking a small step towards this goal, recent work has begun developing artificial agents that follow natural language navigation instructions in perceptually-rich, simulated environments (Anderson et al., 2018; Chen et al., 2019). An example instruction might be "*Go down the hall and turn left at the wooden desk. Continue until you reach the kitchen and then stop by the kettle.*" and agents are evaluated by their ability to follow the described path in (potentially novel) simulated environments.

Many of these tasks have been developed from datasets of panoramic images captured in real scenes – e.g. Google StreetView images in Touchdown (Chen et al., 2019) or Matterport3D panoramas captured in homes in Vision-and-Language Navigation (VLN) (Anderson et al., 2018). This paradigm enables efficient data collection and high visual fidelity compared to 3D scanning or creating synthetic environments; however, scenes are only observed from a sparse set of points relative to the full 3D environment ($\sim$117 viewpoints per environment in VLN). As a consequence, environments in these tasks are defined in terms of a navigation graph (nav-graph) – a static topological representation of 3D space. As shown in Fig. 1(a), nodes in the nav-graph correspond to 360° panoramic images taken at fixed locations and edges between nodes indicate navigability. This nav-graph based formulation introduces a number of assumptions that make it a poor proxy for what a robotic agent would encounter while navigating the real world.

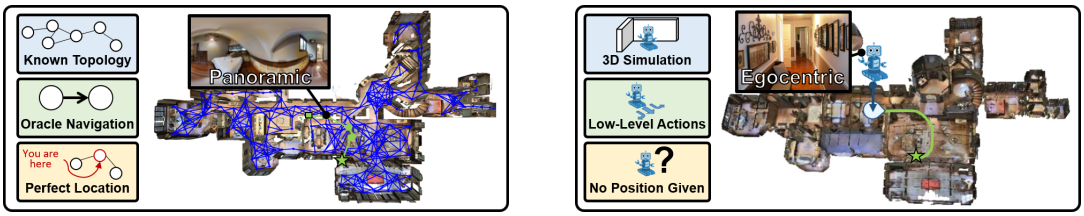
Focusing our discussion on Vision-and-Language Navigation (VLN), the existence and common usage of the nav-graph imply the following assumptions:

- **Known topology.** Rather than continuous environments in which agents can move freely, agents operate on a fixed topology of traversable nodes (shown in blue in Fig. 1(a)). Aside from being a poor match to robot control, this also provides prior information about environment layout to agents – even in "unseen" test settings. How an actual agent might acquire and update such a topology in new environments is an open question.
- **Oracle navigation.** Movement between adjacent nodes in the nav-graph is deterministic, implying the existence of an oracle navigator capable of accurately traversing multiple meters in the presence of obstacles – abstracting away the problem of visual navigation. This is in contrast to the continuous stream of observations a real agent would encounter while moving.
- **Perfect localization.** Agents are given their precise location and heading at all times. Most works use this data to encode precise geometry between nodes in the nav-graph as part of the decision making process, e.g. moving 30°W and 1.12m forward. However, precise localization indoors is still a challenging problem.

These assumptions are often justified by invoking existing technologies as potential oracles. For example, simultaneous localization and mapping (SLAM) or odometry systems

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.



(a) Vision-and-Language Navigation (VLN)

(b) VLN in Continuous Environments (VLN-CE)

Figure 1. The VLN setting **(a)** operates on a fixed topology of panoramic images (shown in blue) – assuming perfect navigation between nodes (often meters apart) and precise localization. Our VLN-CE setting **(b)** lifts these assumptions by instantiating the task in continuous environments with low-level actions – providing a more realistic testbed for robot instruction following.

can offer strong localization in appropriate conditions. Likewise, algorithms for path planning and control can navigate short distances in the presence of obstacles. Further, it is reasonable to suggest that issuing commands at the level of relative waypoints (in analogy to nav-graph nodes) is the proper interface between language-guided AI navigators and lower-level agent control. However, these techniques are each independently far from perfect and such an agent would need to learn the limitations of these lower-level control systems – facing consequences when proposed waypoints cannot be reached effectively. Integrative studies such as these that combine and evaluate techniques for control and mapping with learned AI agents are not possible in current nav-graph based problem settings. In this work, we develop a continuous setting that enables these types of studies and take a first step towards integrating VLN agents with control via low-level actions.

We develop Vision-and-Language Navigation in Continuous Environments (VLN-CE). In VLN-CE, agents in our task are free to navigate to any unobstructed point through a set of low-level actions (e.g. move `forward` 0.25m, `turn-left` 15 degrees) rather than teleporting between fixed nodes. This setting introduces many challenges ignored in prior work. Agents in VLN-CE face significantly longer time horizons; the average number of actions along a path in VLN-CE is ∼55 compared to the 4-6 node hops in VLN. Moreover, the views the agent receives along the way are not well-posed by careful human operators as in the panoramas, but rather a consequence of the agent's actions. Further, agents are not provided their location or heading.

We develop two agent architectures for this task and perform input-modality ablations to assess the biases and baselines in this new setting. Unlike in VLN where depth is rarely used, our analysis reveals depth to be an integral signal for learning embodied navigation. Our best agent successfully navigates to the goal in approximately a third of episodes in unseen environments – taking an average of 88 actions.

VLN-CE is set in the space of language-guided visual navigation where a number of recent tasks have been proposed (Anderson et al., 2018; Chen et al., 2019; Misra et al., 2018; Hermann et al., 2020). The variation in these tasks is primarily in the source of navigation instructions (crowdsourced vs. generated via template), environment realism (hand-designed synthetic worlds vs. captures from real locations), and constraints on agent navigation (nav-graph based navigation vs. unconstrained agent motion).

Task	Instructions	Environment	Navigation
LANI (Misra et al., 2018)	Crowdsourced	Synthetic	Unconstrained
StreetNav (Hermann et al., 2020)	Templated	Real	Nav-Graph Based
Touchdown (Chen et al., 2019)	Crowdsourced	Real	Nav-Graph Based
VLN (Anderson et al., 2018)	Crowdsourced	Real	Nav-Graph Based
VLN-CE (ours)	Crowdsourced	Real	Unconstrained

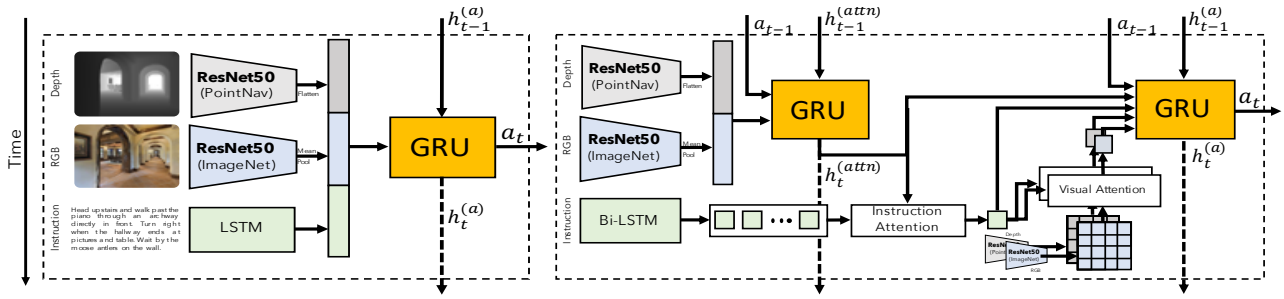
Our proposed VLN-CE task provides the first setting with crowdsourced instructions in realistic environments with unconstrained agent navigation.

2. VLN in Continuous Environments

We consider a continuous setting for the VLN task which we refer to as Vision-and-Language Navigation in Continuous Environments (VLN-CE). Given a natural language navigation instruction, an agent must navigate from a start position to the described goal in a continuous 3D environment by executing a sequence of low-level actions based on egocentric perception alone. In overview, we develop this setting by transferring nav-graph-based Room-to-Room (R2R) (Anderson et al., 2018) trajectories to reconstructed continuous Matterport3D environments in the Habitat simulator (Savva et al., 2019).

Continuous Matterport3D Environments in Habitat. We set our task in the Matterport3D (MP3D) (Chang et al., 2017) dataset, a collection of 90 environments. To enable agent interaction with the 3D mesh reconstructions, we develop the VLN-CE task on top of the Habitat Simulator (Savva et al., 2019). In contrast to the simulator used in VLN (Anderson et al., 2018), Habitat allows agents to navigate freely in the continuous environments.

Observations and Actions. We select observation and action spaces to emulate a ground-based, zero-turning radius robot with a single, forward-mounted RGBD camera. Agents perceive the world through egocentric RGBD images with a resolution of 256×256 and a horizontal field-of-view of 90 degrees. We consider four low-level actions for agents in VLN-CE – move `forward` 0.25m, `turn-left`



(a) Sequence-to-Sequence Baseline (b) Cross-Modal Attention Model

Figure 2. We develop a simple baseline agent (a) as well as an attention-based agent (b) that reflect trends in VLN modeling.

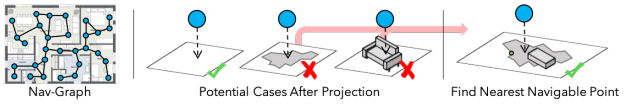


Figure 3. We transfer nav-graph trajectories over panoramas (blue dots) from the Room-to-Room (R2R) dataset to reconstructed Matterport3D (MP3D) environments, adjusting for mesh errors.

or `turn-right` 15 degrees, or `stop` to declare that the goal position has been reached. These actions can easily be implemented on robots with standard motion controllers.

2.1. Transferring Nav-Graph Trajectories

Rather than collecting a new dataset, we instead transfer instruction and trajectories from the nav-graph-based Room-to-Room dataset. Doing so enables us to compare existing nav-graph-based techniques with our methods that operate in continuous environments on the same instructions.

MP3D Simulator and the Room-to-Room Dataset. Anderson et al. (2018) developed the Matterport3D Simulator to enable agent interaction with MP3D panoramas. Environments in this simulator are defined as nav-graphs $E = \{\mathcal{V}, \mathcal{E}\}$. Each node $v \in \mathcal{V}$ corresponds to a panoramic image I at location x, y, z – i.e. $v = \{I, x, y, z\}$. Edges in the graph indicate oracle navigability. Anderson et al. then collect the Room-to-Room (R2R) dataset containing 7189 trajectories each with three human-generated instructions.

Converting Room-to-Room Trajectories to Habitat. Given a mapping between the coordinate frames of Matterport3D Simulator and MP3D in Habitat, it is seemingly simple to transfer the Room-to-Room trajectories – after all, each node has a corresponding xyz location. However, node locations often do not correspond to reachable locations for a ground-based agent – existing at variable height depending on tripod configuration or placed on top of flat furniture like tables. Fig. 3 shows an overview of our conversion process.

For each node, $v = \{I, x, y, z\}$, we identify the closest point on the reconstructed mesh that can be occupied by a ground-based agent. Directly projecting to the nearest mesh location fails for 73% of nodes, either projecting to distant (>0.5m) or non-navigable points. Instead, we perform a vertical ray trace with a limited horizontal displacement. If no navigable point is found, we consider the MP3D node invalid. We reviewed invalid nodes manually and made corrections if possible. After these steps, 98.3% of nodes are successfully transferred which we refer to as waypoints. Finally, we verify that an oracle agent can navigate each trajectory of waypoints $\tau = [w_1, \ldots, w_T]$. In total, we find 77% of the R2R trajectories navigable in the continuous environment. The 23% that were not navigable either included an invalid node or spanned disjoint regions of the reconstruction.

2.2. VLN-CE Dataset

The VLN-CE dataset consists of 4475 trajectories converted from R2R train and validation splits. Each trajectory has the multiple instructions from R2R and a pre-computed shortest path following the waypoints via low-level actions.

3. Instruction-guided Nav Models in VLN-CE

We develop two models for VLN-CE. A simple sequence-to-sequence baseline and a more powerful cross-modal attention model which are conceptually similar to early and more recent work in the nav-graph based VLN task.

Encoding Inputs. We convert tokenized instructions to word embeddings $\mathbf{w}_1, \ldots, \mathbf{w}_T$ for a length T instruction which are then processed by recurrent encoders. We apply a pretrained ResNet50 to RGB observations to collect semantic visual features. Likewise for depth observations, we use a ResNet50 trained in point-goal navigation.

3.1. Sequence-to-Sequence Baseline

We consider a simple sequence-to-sequence baseline model shown in Fig. 2(a) consisting of a recurrent policy that takes visual observations (depth and RGB) and an instruction embedding at each time step, then predicts an action a. This simple model enables straight-forward input-modality ablations and establishes a baseline for the VLN-CE setting.

3.2. Cross-Modal Attention Model

We consider a more expressive model shown in Fig. 2(b) that incorporates mechanisms for cross-modal attention and spa-

Table 1. Performance in VLN-CE. We find that the `Cross-Modal` model outperforms all other models.

					Val-Seen						Val-Unseen					
#	Model	Vision	Instr.	History	TL ↓	NE ↓	nDTW ↑	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	nDTW ↑	OS ↑	SR ↑	SPL ↑
1	Hand-Crafted	-	-	-	3.83	9.56	0.33	0.05	0.04	0.04	3.71	10.34	0.30	0.04	0.03	0.02
2	Seq2Seq-Base	RGBD	✓	✓	8.40	8.54	0.45	0.35	0.25	0.24	7.67	8.94	0.43	0.25	0.20	0.18
3	– No Image	D	✓	✓	7.77	8.55	0.46	0.31	0.24	0.23	7.87	9.09	0.41	0.23	0.17	0.15
4	– No Depth	RGB	✓	✓	4.93	10.76	0.29	0.10	0.03	0.03	5.54	9.89	0.31	0.11	0.04	0.04
5	– No Vision	-	✓	✓	4.26	11.07	0.26	0.03	0.00	0.00	4.68	10.06	0.30	0.07	0.00	0.00
6	– No Instruction	RGBD	-	✓	7.86	9.09	0.42	0.26	0.18	0.17	7.27	9.03	0.42	0.22	0.17	0.16
7	Seq2Seq	RGBD	✓	✓	9.32	**7.09**	0.53	0.44	0.34	0.32	8.46	7.92	0.48	0.35	0.26	0.23
8	Cross-Modal	RGBD	✓	✓	9.26	7.12	**0.54**	**0.46**	**0.37**	**0.35**	8.64	**7.37**	**0.51**	**0.40**	**0.32**	**0.30**

tial visual reasoning. This model consists of two recurrent networks – one tracking visual observations as before and the other tracking attended instruction and visual features.

4. Experiments

Setting and Metrics. We train and evaluate our models in VLN-CE. We report standard metrics for visual navigation tasks defined in (Anderson et al., 2018; Magalhaes et al., 2019) – trajectory length in meters (`TL`), terminal navigation error in meters (`NE`), oracle success rate (`OS`), success rate (`SR`), success weighted by inverse path length (`SPL`), and normalized dynamic-time warping (`nDTW`). For our discussion, we examine SR and SPL as the primary metrics.

4.1. Establishing Baseline Performance for VLN-CE

Non-Learning Baseline. Shown in Tab. 3 (row `1`), we evaluate a `hand-crafted` agent that picks a random heading and takes 37 `forward` actions (average trajectory length) before calling `stop`. It achieves a 3% success rate in val-unseen. In contrast, a similar hand-crafted random-heading-and-forward model in VLN yields a 16.3% success rate (Anderson et al., 2018). This gap illustrates the strong structural prior provided by the nav-graph in VLN.

Seq2Seq and Single-Modality Ablations. Tab. 3 also shows performance for the baseline Seq2Seq model along with single-modality ablations as trained with imitation learning (rows `2-6`). `Seq2Seq-Base` outperforms the hand-crafted baseline and all modality ablations with a 20% val-unseen success rate. We find that depth is a very strong signal for learning, with models lacking it (`No Depth` and `No Vision`) failing to outperform chance. We believe that depth enables agents to quickly begin traversing environments effectively (e.g. without collisions) and without this it is very difficult to bootstrap to instruction following.

4.2. Model Performance in VLN-CE

We then use an enhanced training regime for both `Seq2Seq` and `Cross-Modal` models. First, we find DAgger training increases `Seq2Seq` performance from 0.18 SPL to 0.23 SPL (rows `2` vs. `7`). Next, we find that while not effective for the `Seq2Seq` model, data-augmentation (Tan et al., 2019) and progress monitor (Ma et al., 2019) loss (in addition to DAgger) improve the `Cross-Modal` model. With these enhancements the `Cross-Modal` model performed the best in unseen environments at 0.30 SPL (a 30% relative improvement over `Seq2Seq`, rows `7` and `8`).

References

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1, 2, 3, 4

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. 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

Chen, H., Suhr, A., Misra, D., Snavely, N., and Artzi, Y. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*, 2019. 1, 2

Hermann, K. M., Malinowski, M., Mirowski, P., Banki-Horvath, A., Anderson, K., and Hadsell, R. Learning to follow directions in street view. *AAAI*, 2020. 2

Ma, C.-Y., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., and Xiong, C. Self-monitoring navigation agent via auxiliary progress estimation. *ICLR*, 2019. 4

Magalhaes, G., Jain, V., Ku, A., Ie, E., and Baldridge, J. Effective and general evaluation for instruction conditioned navigation using dynamic time warping. *arXiv:1907.05446*, 2019. 4

Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y. Mapping instructions to actions in 3d environments with visual goal prediction. In *EMNLP*, 2018. 2

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. Habitat: A Platform for Embodied AI Research. *ICCV*, 2019. 2

Tan, H., Yu, L., and Bansal, M. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL HLT*, 2019. 4