Appendix: Bridging the Gap Between Learning in Discrete and Continuous Environments for Vision-and-Language Navigation

Yicong Hong[1*] Zun Wang[1*] Qi Wu[2] Stephen Gould[1]
[1]The Australian National University, [2]University of Adelaide
{yicong.hong, zun.wang, stephen.gould}@anu.edu.au
qi.wu01@adelaide.edu.au

Project URL: https://github.com/YicongHong/Discrete-Continuous-VLN

A. Connectivity Graph

We detail the construction of connectivity graphs in Habitat-Matterport3D environments, as well as the adjustments on the R2R-CE trajectories.

A.1. Graph Construction (§4.2[1])

Projection We start with the pre-defined connectivity graphs in MP3D environments, and leverage trajectories in datasets to adjust the position of nodes. For each MP3D scene, the corresponding graph that contains a set of nodes is projected to the same scene in Habitat. Note that, each node that is applied by VLN-CE [12], for creating the continuous ground-truth paths, is projected to the averaged position of points on the ground-truth paths that are closest to this node. However, such projection only fixes a small portion of invalid nodes (*i.e.* nodes within obstacles) and edges (*i.e.* edges intersects with obstacles). To obtain a fully navigable graph, we design a heuristic to further adjust the position of inaccessible nodes in the environment.

Criteria Four criteria are followed to ensure the quality of the resulting graphs: (1) Nodes should not adhere to obstacles. (2) As few as number of nodes should be added for correcting invalid edges. (3) Straightness of the edges should be maintained. (4) Nodes connected with an edge shorter than 0.25 meters should be merged.

Sampling For each node that requires an adjustment, we sample 264 points within a 0.35m-radius circle centered at the node as its candidate new positions. Those candidates are uniformly sampled on several rings evenly at `(0.1,18)`, `(0.15,30)`, `(0.2,36)`, `(0.25,48)`, `(0.3,60)` and `(0.35,72)`, where `(radius,#samples)`. According to the aforementioned *Criteria*, for all candidate positions that are not within an obstacle, we score them by a weighted sum of three measurements: (1) distance to closest obstacles, (2) number of new nodes that needs to be added, and (3) edge straightness which is computed as the ratio between the sum of lengths of new edges and the original edge length. The candidate position which has the highest score will become the new position of the node.

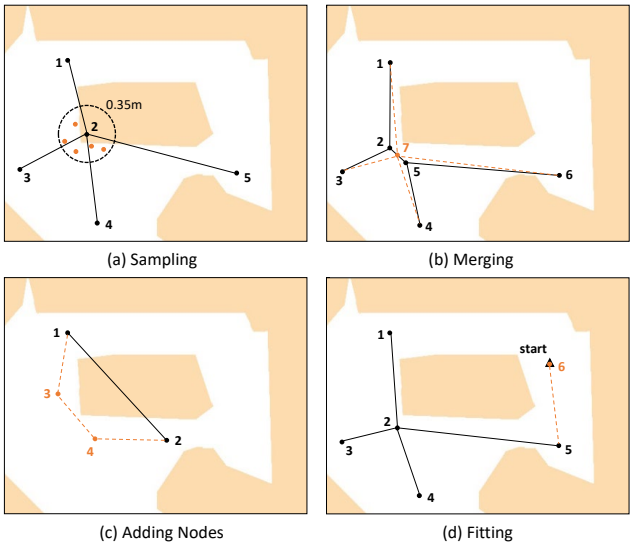


Figure 1. Graph construction methods. (a) The invalid node **2** is adjusted to a newly sampled position in a 0.35m circle. (b) Node **2** and node **5** are merged to a new node **7**, the edges are adjusted accordingly. (c) An invalid edge between nodes **1** and **2** is replaced by a detour and two new nodes (**3** and **4**) are added to the graph. (d) New nodes (**6**) are added to the graph if there is no adjacent node to the starting or ending position of a trajectory

Adding Nodes Detours are sometimes necessary for addressing the invalid edges that go through obstacles, in this

[1]Link to Section 4.2 in Main Paper.

case, new nodes are added to the graph. In specific, we apply the `shortest_path_follower` in the *Habitat Simulator* [15] to generate numbers of paths with various step lengths that connect the two nodes. Nodes within these paths are evaluated using the *Criteria* so that the best set of new nodes can be determined and added to the graph.

Merging Once a fully navigable graph is created, nodes within 0.5m are merged to a single node to make the graph more concise. However, this may creates new invalid nodes and edges. As a result, we repeat the *Adjustment*, *Adding Nodes* and *Merging* processes until the entire connectivity graph is fully navigable with edges longer than or equals to 0.5 meters.

Fitting Finally, to match the R2R-CE data [2, 12], we fit the graph for the starting and ending positions of all trajectories in the dataset (except for the *test* split). We check if all the starting and ending points can be matched to nodes on graphs within a geodesic distance of 0.8 meters, if not, new nodes will be added to connect the graph to those positions (using our *Adding Nodes* method).

Visualization Visualizations of the resulting Habitat-MP3D graphs and the comparison to the original MP3D graphs can be found in Figure 2 and Figure 3.

A.2. R2R-CE Trajectories Adjustment (§5.1)

To establish a fair comparison between discrete and continuous navigation in Habitat (§5.1), we re-compute and filter the original R2R-CE trajectories based on our Habitat-MP3D graph. Specifically, for each continuous trajectory in R2R-CE [12], we collect the three closest nodes to its starting and the ending points, respectively. Then, for each one of the nine pairs of starting and ending nodes, we collect up to 200 shortest discrete paths on graph from all the possible paths that connect the two nodes. For the resulting 1,800 paths, we measure the nDTW [11] between each path and the original continuous trajectory, and select the path with the highest score as the discrete ground-truth path on the Habitat-MP3D graph. Samples which have a discrete trajectory with nDTW lower than 0.92 will be discarded from the dataset. Such expensive and strict filtering process ensures that only samples with well-matched discrete and continuous paths remain in the dataset, which is suitable for us to compare navigation with high-level and low-level actions.

Overall, we obtain 10,755, 1,755 and 745 episodes on the train, validation seen, and validation unseen split, respectively. For reference, the original R2R-CE [12] has 10,819, 1,839 and 778 episodes in the data splits. Besides, the averaged number of nodes on a trajectory is 6.00 and 6.63 for samples in R2R [2] and in our filtered dataset, respectively.

A.3. Agent Performance (§5.2)

As shown in Table 1, we experiment with the same agents on MP3D graphs and on our Habitat-MP3D graphs. Surprisingly, despite the fact that agents receive lower-quality images and navigate on a more complicated graph in Habitat, similar performance are achieved in unseen environments.

Model	Val	MP3D R2R				Habitat-MP3D R2R			
		TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑
CMA	S	15.20	4.29	54.95	49.67	10.21	6.28	41.48	36.58
	U	19.68	5.54	39.97	33.36	10.47	6.51	39.32	33.89
VLN↻BERT	S	16.62	4.00	54.65	46.58	14.21	4.63	52.21	42.53
	U	16.75	4.59	51.13	42.92	14.34	5.22	48.89	40.36

Table 1. Performance of agents navigating on MP3D and Habitat-MP3D graphs in Seen (*S*) and Unseen (*U*) environments.

B. Navigator Networks

In this section, we provide the implementation details of the Cross-Modal Matching agent (CMA) [18] and the Recurrent VLN-BERT Agent (VLN↻BERT) [10] on the R2R-CE [2] and the RxR-CE [13] datasets.

B.1. Architecture (§5.1)

Visual Encoders Agents in our experiments apply the same RGB and depth encoders to process the candidate images at waypoint directions. As in VLN-CE [12], we use two ResNet-50 [8], one pre-trained on ImageNet [14] for classification and another one pre-trained on Gibson [20] for point-goal navigation [19], to encode the RGB and depth inputs, respectively. These encoders are freezed while training the navigators, where the outputs are fed into the candidate waypoints predictor to infer adjacent waypoints, and into the navigators for *view selection*.

Refer to the *Main Paper* §3, we denote the encoded representations as $\{\boldsymbol{v}_1^{rgb}, \boldsymbol{v}_2^{rgb}, \dots, \boldsymbol{v}_k^{rgb} \mid \boldsymbol{v}_i^{rgb} \in \mathbb{R}^{2048}\}$ and $\{\boldsymbol{v}_1^{d}, \boldsymbol{v}_2^{d}, \dots, \boldsymbol{v}_k^{d} \mid \boldsymbol{v}_i^{d} \in \mathbb{R}^{128}\}$[2], corresponding to the k directions with waypoints. The RGB and depth representations are merged before passing to the policy networks as

$$\boldsymbol{f}_i = \left[\boldsymbol{v}_i^{rgb}\boldsymbol{W}_{rgb}; \boldsymbol{v}_i^{d}\boldsymbol{W}_{depth}; \boldsymbol{d}_i\right]\boldsymbol{W}_{merge} \qquad (1)$$

where $\boldsymbol{W}$ are learnable linear projections with ReLU activation. Following previous works [6, 16], we explicitly encode the relative direction of each candidate view as $\boldsymbol{d}_i$. $\boldsymbol{d}_i$ is a vector formed by replicating $(\cos\theta_t^i, \sin\theta_t^i)$ by 32 times, where θ_t^i is the heading angle of the view with respect to the agent's orientation. Note that, unlike previous works, $\boldsymbol{d}_i$ in our experiments does not involve an elevation angle because the waypoints predictor only creates a 2D-planar graph. We suggest that predicting 3D waypoints could be a

[2] Subscript t for time step is omitted here for simplicity.

valuable extension for future work. Finally, the overall visual feature $\boldsymbol{f}_i$ is of dimension 512 and 768 for the CMA and the VLN↻BERT (to match the default hidden dimension of V&L BERT [7, 10]) models, respectively.

Language Encoders and Initial States Given a natural language instruction $\boldsymbol{U}$ of a sequence of l words $\langle w_1, w_2, \ldots, w_l \rangle$, the agents first encode the instruction into textual representations. For CMA, a bidirectional LSTM [9] is applied to encode $\boldsymbol{U}$ with randomly initialized word embeddings as

$$\boldsymbol{X} = \langle \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_l} \rangle = \text{Bi-LSTM}\left(w_1, w_2, \ldots, w_l\right) \quad (2)$$

For VLN↻BERT, the language stream of the two-stream V&L transformers [7] is applied to process $\boldsymbol{U}$ as

$$\boldsymbol{s}_0, \boldsymbol{X} = \text{VLN↻BERT}(\texttt{[CLS]}, \boldsymbol{U}, \texttt{[SEP]}) \quad (3)$$

with word embeddings pre-trained in PREVALENT [7]. `[CLS]` and `[SEP]` are the classification token and the separation token pre-defined in BERT [5]. VLN↻BERT adopts the output of the `[CLS]` token $\boldsymbol{s}_0$ as the agent's initial state, whereas CMA uses a vector of zeros to represent the initial state.

CMA We apply the original CMA [18] as the policy network in our experiments, which is different from the CMA implemented in VLN-CE [12] that has an additional recurrent module. In specific, at each navigation step, the agent's state is encoded as:

$$\boldsymbol{s}_t = \text{GRU}\left(\left[\boldsymbol{c}_t^{vis,0}; \boldsymbol{a}_{t-1}\boldsymbol{W}_{act}\right], \boldsymbol{s}_{t-1}\right) \quad (4)$$

where $\boldsymbol{a}_{t-1}$ is the agent's past decision represented by the aforementioned directional encoding and $\boldsymbol{W}_{act}$ is a learnable projection with a hyperbolic tangent activation. $\boldsymbol{c}_t^{vis,0} = \text{SoftATTN}(\boldsymbol{f}_t, \boldsymbol{s}_{t-1})$ is the weighted sum of the candidate features $\boldsymbol{f}_t$, where the weights are produced by a dot-product based soft-attention [17] using the agent's past state $\boldsymbol{s}_{t-1}$ as query. Then, the current state $\boldsymbol{s}_t$ is applied to compute the updated textual and visual features as $\boldsymbol{c}_t^{lang} = \text{SoftATTN}(\boldsymbol{X}, \boldsymbol{s}_t)$ and $\boldsymbol{c}_t^{vis,1} = \text{SoftATTN}(\boldsymbol{f}_t, \boldsymbol{s}_t)$. Finally, the probability of each candidate directions is computed as

$$p_{t,i} = \text{Softmax}\left(\left[\boldsymbol{s}_t; \boldsymbol{c}_t^{vis,1}; \boldsymbol{c}_t^{lang}\right]\boldsymbol{W}_a\left(\boldsymbol{f}_{t,i}\boldsymbol{W}_b\right)^T\right) \quad (5)$$

In inference, the agent will select the view with the greatest probability and navigate to the waypoint in that view.

VLN↻BERT We apply the PREVALENT variant [7] of the VLN↻BERT [10] with slight modifications in our experiments. Specifically, we remove the *cross-modal matching* in *state refinement* since the method complicates the network while leading to a trivial improvement. VLN↻BERT applies a multi-layer transformers to perform cross-modal soft-attention across the agent state, encoded language and visual features:

$$\boldsymbol{s}_t, \boldsymbol{p}_t = \text{VLN↻BERT}(\boldsymbol{s}_{t-1}^d, \boldsymbol{X}, \boldsymbol{f}_t) \quad (6)$$

where $\boldsymbol{s}_{t-1}^d = [\boldsymbol{s}_{t-1}; \boldsymbol{a}_{t-1}]\boldsymbol{W}_{act}$ is the previous state with directional encoding, $\boldsymbol{p}_t$ is the action probabilities computed as the mean attention weights of the visual tokens $\boldsymbol{f}_t$ over all the attention heads in the last transformer layer with respect to the state.

B.2. Training (§5.1)

All agents in our experiments are trained using imitation learning (IL) with a cross-entropy loss on the action probabilities p_t and the oracle action a_t^* as $\mathcal{L}_{IL} = -\sum_t a_t^* \log\left(p_t\right)$ where the oracle action is to the waypoint which has the shortest geodesic distance to the target, among all predicted waypoints. The oracle stop is positive when the geodesic distance between the agent and the target is shorter than 1.5 meters. Note that, in some rare cases (Figure 5), the candidate waypoints predictor might not be able to return a waypoint that brings the agent closer to the target[3]. In order to allow the agent to explore the environment while learning from teacher actions, we control the agent with schedule sampling [3] to sample an action between oracle and prediction at each step.

Oracle Actions in RxR-CE Each path in RxR-CE [13] is composed of multiple shortest sub-paths traversing through a sequence of rooms, which is not necessarily the shortest path to the target. As a result, we design a different method to determine the oracle actions: At each time step, we compute a sub-goal as the intersection of a 3-meter ring centered at the agent and the ground-truth path. Then, the oracle action is to the waypoint which has the shortest sum of geodesic distances from the agent to the waypoint and from the waypoint to the sub-goal. If there is more than one intersection, we use the one that is the farthest on the ground-truth path as the sub-goal. If there is no intersection, we apply the latest sub-goal as the current sub-goal to push the agent to return to the ground-truth path.

Initialization The VLN↻BERT is initialized from the pre-trained PREVALENT model [7]. When training on RxR-CE [13], we apply the multilingual BERT features to initialize the word embeddings in both the CMA and VLN↻BERT. We train each model three times each for a

[3] We experiment with taking the oracle action at each step for samples in unseen environments in the original VLN-CE data, results show that only 2% of agents cannot reach within 3 meters to the target.

different language and combine the results at the end for submitting to the test server.

Hardware and Time Cost On R2R-CE, we train the CMA and the VLN↻BERT for 50 epochs with a batch size of 16, both models take about 3.5 days to complete using a single NVIDIA RTX 3090 GPU. On RxR-CE, we train the models for 25 epochs on a single GPU, with a batch size of 16 and 8 for the CMA and the VLN↻BERT, respectively. The training takes about 3 days per language due to the multilingual and larger dataset, and longer paths.

B.3. Evaluation Metrics (§3.1)

Trajectory Length (TL): the average navigation path length in meters, Navigation Error (NE): the average distance between the target and the agent's final position in meters, Success Rate (SR): the ratio of stopping within 3 meters to the target, Success weighted by the normalized inverse of the Path Length (SPL) [1], normalized Dynamic Time Warping (nDTW) and Success weighted by normalized Dynamic Time Warping (SDTW) [11]. While SR focuses on the accuracy of agent's decisions, SPL measures if the agent navigates efficiently, nDTW and SDTW measure if the agent follows the given instructions by computing the similarity between the executed path and the ground-truth path.

C. Visualization (§5.2)

Figure 4 and Figure 5 visualize the agent's trajectories and the predicted waypoints at each step. As shown by the waypoints (indicated with cylinders) in panoramas and the occupancy maps, most of the predictions are nicely positioned at accessible spaces, pointing towards explorable directions around the agent. Thanks to the predicted waypoints, the agent often only needs to make a few decisions for completing a long navigation task. However, in some cases, *e.g.* Figure 5.(Right); the agent is not able to explore certain part of the environment when the predictor fails to produce a waypoints, which disturbs the training and inhibits the navigation. We suggest that future work to utilize the online structural information (*e.g.* depth) and agent's behavior, or to make the predictor to collaborate with the agent for producing more flexible waypoints.

D. Simulator Configurations (§5.1)

According to the official configurations[4], agents in R2R-CE [2, 12] and in RxR-CE [13] are set up differently in Habitat [15]. For example, agent in RxR-CE is shorter in height so that it can access places in the environments where an R2R-CE agent cannot reach. For a fair comparison to previous works, agents in our experiments are set up with the standard dimensions for R2R-CE and RxR-CE, respectively. On the other hand, to facilitate the use of the same waypoints predictor and the powerful pre-trained depth-encoder [19] in the two datasets, as well as to decrease the rendering cost, we adjust the camera parameters in RxR-CE. Some key configurations are listed here, where the commented numbers are the original values.

R2R-CE Configurations:

```
FORWARD_STEP_SIZE: 0.25
TURN_ANGLE: 3x   #15
AGENT:
  HEIGHT: 1.50
  RADIUS: 0.10
HABITAT_SIM:
  ALLOW_SLIDING: True
RGB_SENSOR:
  WIDTH: 224
  HEIGHT: 224
  HFOV: 90
DEPTH_SENSOR:
  WIDTH: 256
  HEIGHT: 256
  HFOV: 90
```

RxR-CE Configurations:

```
FORWARD_STEP_SIZE: 0.25
TURN_ANGLE: 3x   #30
AGENT:
  HEIGHT: 0.88
  RADIUS: 0.18
HABITAT_SIM:
  ALLOW_SLIDING: False
RGB_SENSOR:
  WIDTH: 224   #640
  HEIGHT: 224   #480
  HFOV: 90   #79
DEPTH_SENSOR:
  WIDTH: 256   #640
  HEIGHT: 256   #480
  HFOV: 90   #79
```

`3x` indicates that the step-wise turning angle is a value in $(0°, 3°, \ldots, 357°)$, according to the angular precision of the waypoints predictor.

Turning Angles We would like to point out that a `TURN_ANGLE` of 30° is likely to be unreasonable in the official RxR-CE configurations because such a large turning

[4]R2R-CE code: https://github.com/jacobkrantz/VLN-CE, RxR-CE code: https://github.com/jacobkrantz/VLN-CE/tree/rxr-habitat-challenge.

angle will prevent the agent from heading to certain navigable directions. Moreover, since the ground-truth continuous paths (and the step-wise ground-truth actions) in RxR-CE are generated using a 30° turning angle, it results in a large number of zigzag steps while the agent should walks a straight line[5]. As a result, the original ground-truth paths are inappropriate for imitation learning or for evaluating SPL and nDTW. On the other hand, our waypoints predictor enables an efficient imitation learning without using ground-truths computed with a pre-defined fixed angle.

Sliding Unlike R2R-CE, agents in RxR-CE are not allowed to slide along obstacles on collision, which drastically increases the difficulty of the task and results in agents that can easily fall into deadlocks. To address this issue, we first constrain the vast majority of predicted waypoints to be positioned in open space by stopping augmenting waypoints during training, so that an agent has less probability of hitting obstacles. Moreover, we implement a simple function which runs some test steps to check for navigability around the agent when it is stuck at the same position, and assists the agent to escape from deadlocks.

Model	No Sliding Unseen RxR				Sliding Unseen RxR			
	NE↓	SR↑	SPL↑	nDTW↑	NE↓	SR↑	SPL↑	nDTW↑
CMA	8.76	26.59	22.16	47.05	8.08	31.36	26.93	47.36
VLN↻BERT	8.98	27.08	22.65	46.71	7.62	33.25	28.45	47.57

Table 2. Agents performance with and without sliding.

Table 2 compares the navigation performance in RxR-CE with and without allowing the agents to slide along obstacles on collision. For each model, we train three monolingual agents independently, the results are averaged and presented in the table. By enabling sliding (in which case the waypoint augmentation can be applied), both the CMA and the VLN↻BERT achieve significantly better results.

E. Limitations (§4.1 & §5.2)

In this section, we would like to share some limitations of our proposed method, *i.e.* the candidate waypoints predictor. Although our experiments demonstrate the high accuracy and the strong generalization ability of the module, we believe this information will shine some light on potential room for improvement and greatly benefit future research following this work.

Number of Candidates We limit the maximum number of predicted waypoints to be 5 at any position to reduce the computational cost and stabilize the training of the agents. However, there exists some spatial structures where a position can lead to more navigable directions. We suggest future work to consider a dynamic number of waypoints at different positions.

Prediction at Rare Structures As shown in Figure 5, at some rare structures in the environment such as stairs, the candidate waypoints predictor fails to produce a waypoint on stairs and therefore inhibits agent's navigation. The main reason behind this issue is the amount of training samples for the predictor is insufficient at those structures. Future work can identify rare structures to sample more data and improve the loss function to balance the learning.

Online Prediction Adjustment Our agent fully trusts and applies the predicted waypoints to navigate, which results in issues mentioned above. This problem is more severe in RxR-CE as the agents can easily enter deadlocks but the predicted waypoints cannot help the agents to escape from them. We suggest future work to equip the agent with the ability to adjust waypoints according to the local structure or the agent's control outcomes.

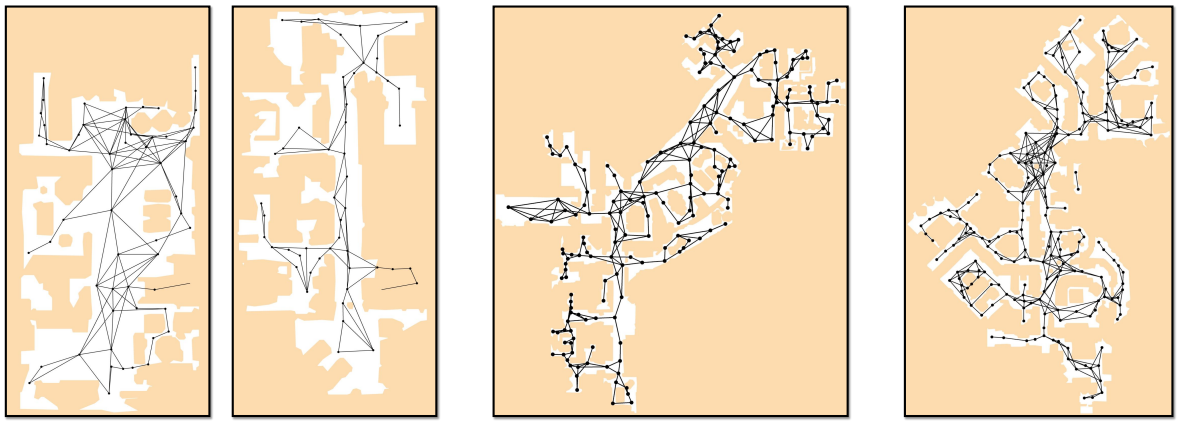
Update Waypoints Predictor Although our candidate waypoints predictor demonstrates high accuracy in unseen MP3D [4] environments, it is unclear if it can be transferred to other distinct and out-of-domain scenes. It would be valuable to develop a waypoints predictor which can be updated to adapt new environments (even without the predefined connectivity graphs).

State-Conditioned Waypoints Prediction In this work, we argue that decoupling the waypoint prediction and agent's decision making can reduce agent's state space and facilitates learning. However, we also recognize that the state information such as the navigation progress and landmarks in instruction could benefit the waypoints predictor for creating more effective waypoints to reach the target. Future work could combine the two ideas to build a waypoints predictor that promotes the navigation.

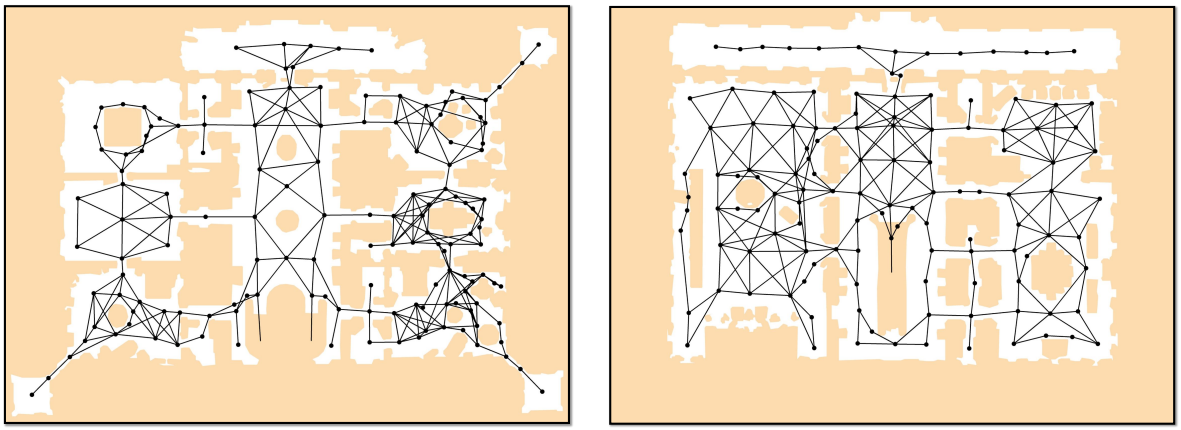
F. Societal Impact

In this research, we apply the R2R-CE [2] and the RxR-CE [13] datasets, which are available under license from Matterport3D [4]. The datasets contain thousands of photos of indoor environments and instructions in different languages for traversing the environments; None of the photos contain recognizable individuals and none of the instructions contain inappropriate language. Experiments are safely and confidentially performed in the Habitat Simulator [15]. This research is at the early stages of pushing towards embodied AI that follows human instructions, there are minimal ethical, privacy or safety concerns. In the future, if a such robot is implemented in real-world, it could benefit the society by assisting people with daily work.

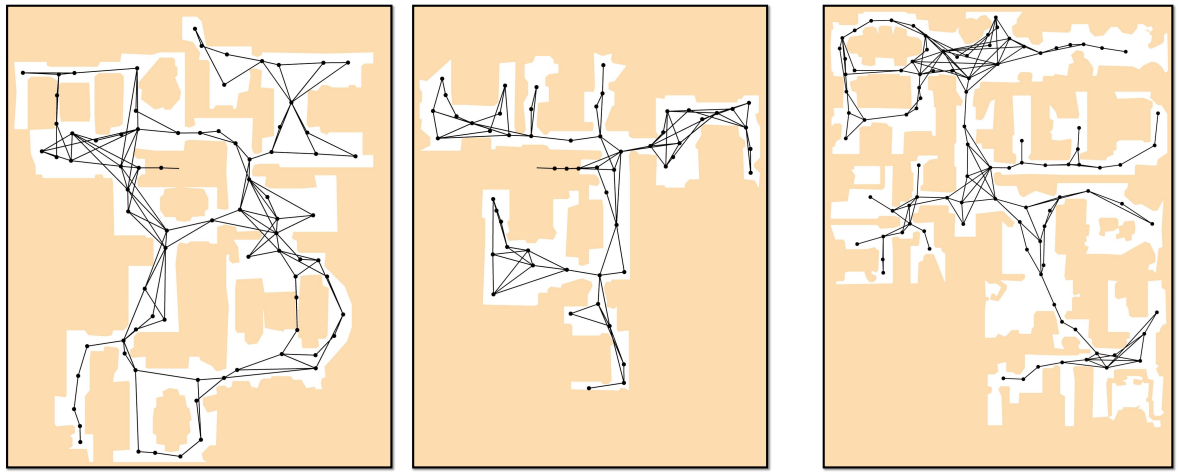
[5]The inflection coefficient of steps is only 1.9 for trajectories in RxR-CE, suggesting that the agent changes actions unreasonably frequently. This number is about 3.2 in R2R-CE when using a 15° turning angle.



Environment A, floor 1-2 Environment B, floor 1 Environment C, floor 1



Environment D, floor 1-2



Environment E, floor 1-2 Environment F, floor 1

Figure 2. Our Habitat-MP3D graphs.

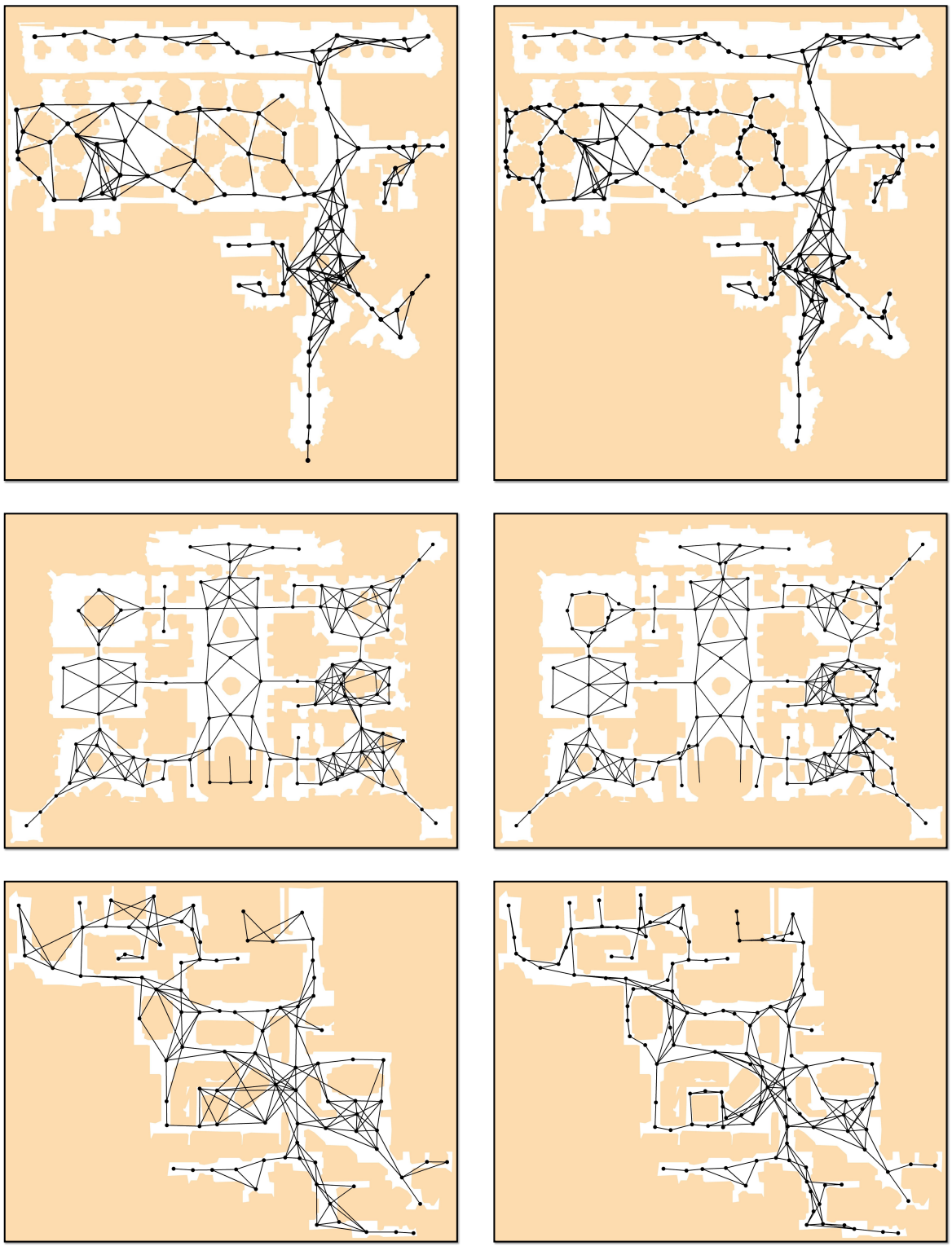


Figure 3. MP3D graphs versus our Habitat-MP3D graphs.

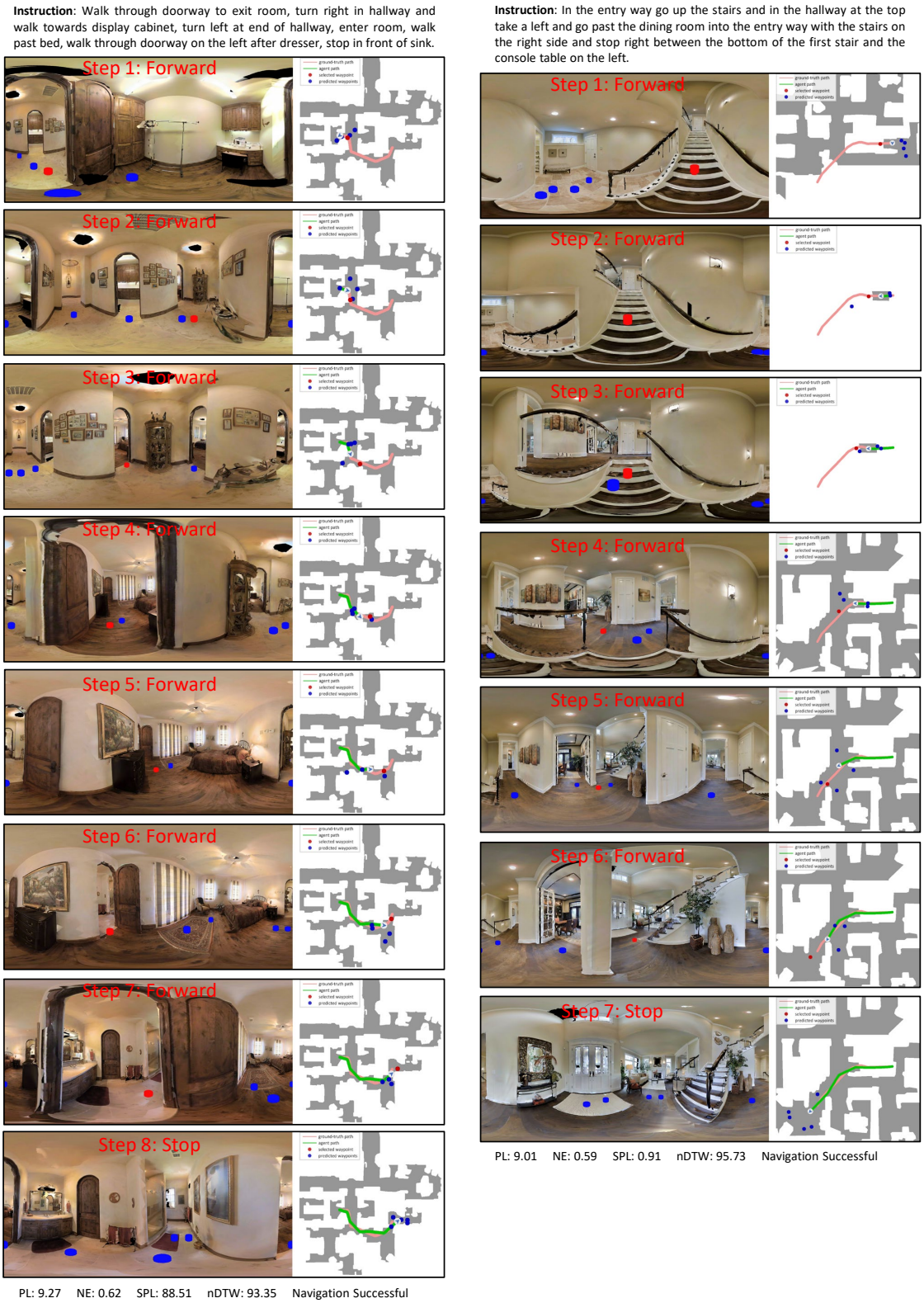


Figure 4. Visualization of trajectories and predicted waypoints. Both samples are successful cases in unseen environments.

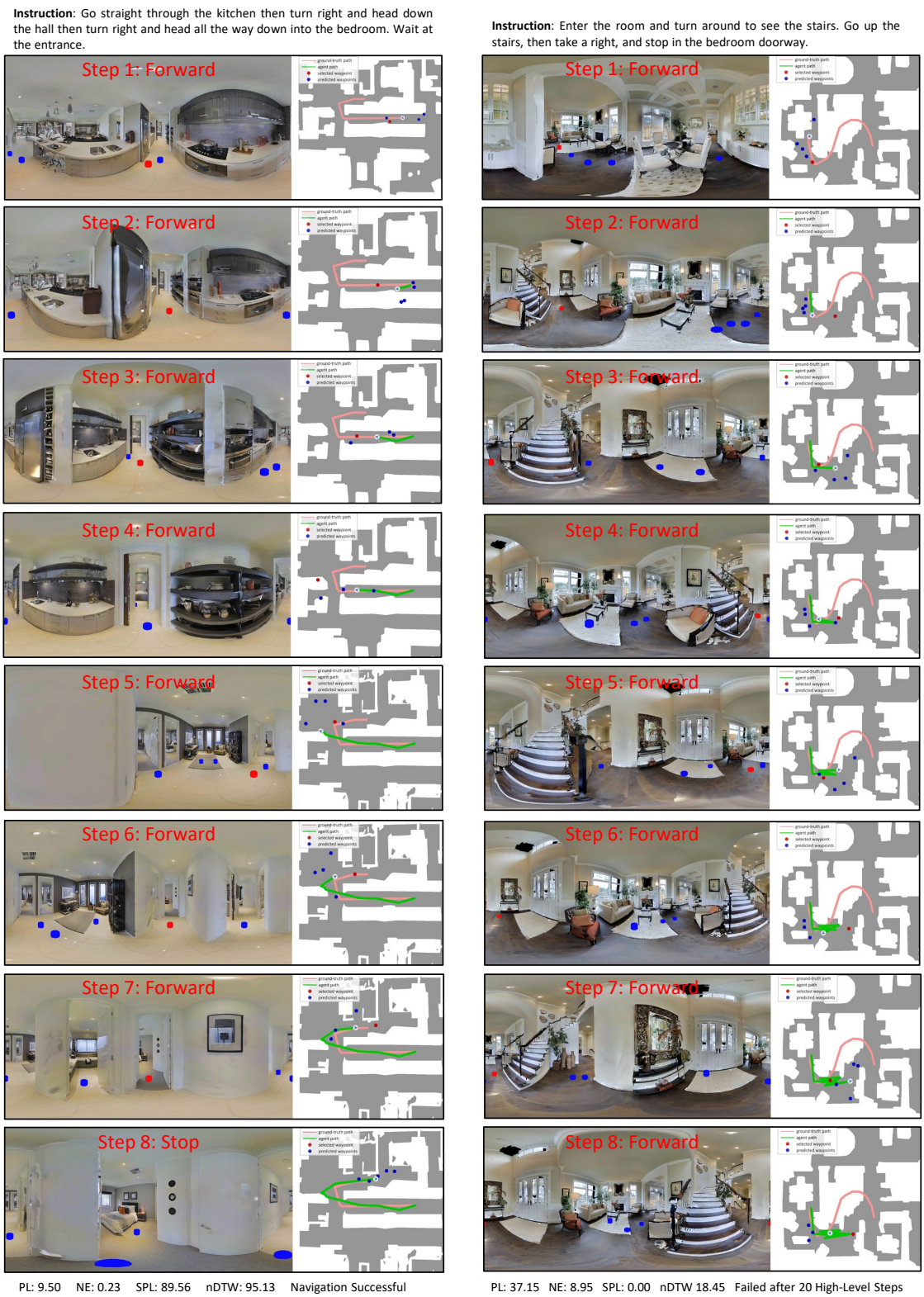


Figure 5. Visualization of trajectories and predicted waypoints. Left: The waypoints predictor produces inaccessible waypoints during navigation (step 2), but the agent avoids choosing those waypoints and eventually reaches the target. Right: The waypoints predictor fails to provide a waypoint on the staircase, so that the agent wanders around the landing and unable to reach the target.

References

[1] 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. 4

[2] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 2, 4, 5

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1171–1179, 2015. 3

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE, 2017. 5

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. 3

[6] 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 *Advances in Neural Information Processing Systems*, pages 3314–3325, 2018. 2

[7] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146, 2020. 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[10] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1643–1653, June 2021. 2, 3

[11] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *arXiv preprint arXiv:1907.05446*, 2019. 2, 4

[12] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, 2020. 1, 2, 3, 4

[13] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4392–4412, 2020. 2, 3, 4, 5

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2

[15] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019. 2, 4, 5

[16] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of NAACL-HLT*, pages 2610–2621, 2019. 2

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[18] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019. 2, 3

[19] 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 *International Conference on Learning Representations*, 2019. 2, 4

[20] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018. 2