

OVER-NAV: Elevating Iterative Vision-and-Language Navigation with Open-Vocabulary Detection and StructurEd Representation

Ganlong Zhao^{1,2} Guanbin Li^{2,3*} Weikai Chen⁴ Yizhou Yu^{1*}

¹The University of Hong Kong ²Sun Yat-sen University

³GuangDong Province Key Laboratory of Information Security Technology

⁴Digital Content Technology Center, Tencent Games

zhaogl@connect.hku.hk, liguanbin@mail.sysu.edu.cn, chenwk891@gmail.com, yizhouy@acm.org

Abstract

Recent advances in Iterative Vision-and-Language Navigation (IVLN) introduce a more meaningful and practical paradigm of VLN by maintaining the agent’s memory across tours of scenes. Although the long-term memory aligns better with the persistent nature of the VLN task, it poses more challenges on how to utilize the highly unstructured navigation memory with extremely sparse supervision. Towards this end, we propose OVER-NAV, which aims to go over and beyond the current arts of IVLN techniques. In particular, we propose to incorporate LLMs and open-vocabulary detectors to distill key information and establish correspondence between multi-modal signals. Such a mechanism introduces reliable cross-modal supervision and enables on-the-fly generalization to unseen scenes without the need of extra annotation and re-training. To fully exploit the interpreted navigation data, we further introduce a structured representation, coded Omnigraph, to effectively integrate multi-modal information along the tour. Accompanied with a novel omnigraph fusion mechanism, OVER-NAV is able to extract the most relevant knowledge from omnigraph for a more accurate navigating action. In addition, OVER-NAV seamlessly supports both discrete and continuous environments under a unified framework. We demonstrate the superiority of OVER-NAV in extensive experiments.

1. Introduction

Vision-and-Language Navigation (VLN) [1] aims to build intelligent agents that can follow natural language instructions to navigate in the unseen environments. However, existing VLN benchmarks eliminate the agent’s memory

*Corresponding authors are Guanbin Li and Yizhou Yu. This work was supported in part by the National Natural Science Foundation of China (NO. 62322608), in part by the CAAI-MindSpore Open Fund, developed on OpenI Community.

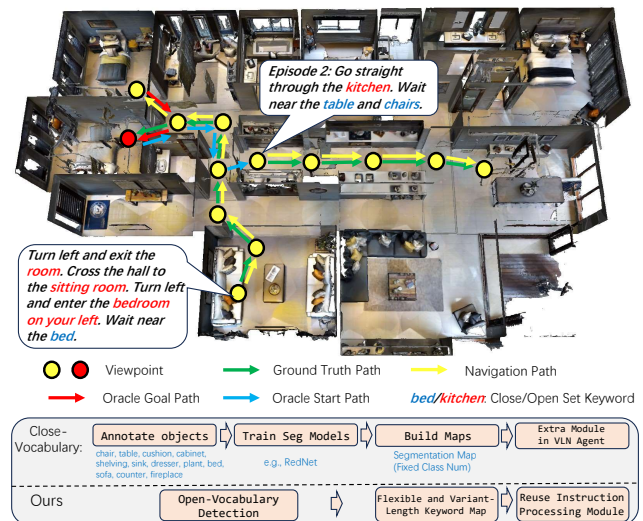


Figure 1. Top: Example of a two-episode tour. The agent first navigates the environment following the instruction of episode 1 (Yellow). Then the agent is directed to the ground truth goal as Oracle Goal phase (Red). Later the agent travels to the start point of episode 2 in the Oracle Start phase (Blue). Finally, the agent navigates the environment following the next instruction (Yellow). Bottom: Comparison between previous methods and ours. Close-vocabulary methods require extra annotation and training efforts to provide segmentation results in navigation, and the agent is limited to a close set of categories building segmentation maps. Our method proposed an open-vocabulary-based omnigraph which is more flexible for various keywords and circumstances.

upon the start of every episode, failing to leverage the visual observations and the iteratively built maps collected by the physical robots. Recent work on Iterative Vision-and-Language Navigation (IVLN) [25] introduces a more meaningful and practical paradigm that orders the episodic tasks in VLN as tours and allows the agents to utilize memory to achieve better navigation performance.

We demonstrate a typical procedure of an IVLN task in

Figure 1. An IVLN agent receives a ordered sequence of instructions that fulfill a tour of the target scene. Each tour consists of individual episodes, each guided by a language instruction. When the agent completes the navigation of an instruction, it is teleoperated by an oracle to the correct goal location, and then translated to the start of the next episode. Hence, each episode is composed of three parts: navigation, oracle goal, and oracle start. In the navigation phase, the agent receives the environment observations along the path P_I following the given instruction I . In the oracle goal/start phase, no instruction is present except the observations along the oracle start path P_{OS} and the oracle goal path P_{OG} . Thus the tour history of each episode is represented as $\{P_{OS}, (I, P_I), P_{OG}\}$.

Although the memory-retaining strategy aligns well with the persistent nature of the VLN tasks, it poses additional challenges on how to fully exploit the unstructured navigation history of previous episodes. First, it remains formidable to interpret the multi-modal information that spans a number of domains including language (instruction), vision (visual measurements), time, and location (physical movement), without any explicit supervision. The only weak supervision existing in the tour history is the correspondence between the instruction I and the navigation path P_I . However, such correspondences are coarse and ambiguous as the visual observations along the path may not correspond to the order of the objects and actions appeared in the instruction. Moreover, P_I could deviate from the correct path due to the erroneous decisions made along the navigation. Therefore, obtaining reliable supervision over the navigation history and establishing faithful correspondence between the multi-modal data are the keys to the success of IVLN tasks.

In the ideal scenarios where all the multi-modal data can be thoroughly interpreted, the second challenge arises in structurizing the extensive memory so that it can be effectively utilized under the IVLN framework. Prior works [25] show that a naive stacking of the history data for feature representation would lead to inferior performance. Additionally, a well structured memory should be general enough to accommodate varying settings in IVLN tasks, e.g. the discrete [1, 25] and continuous environments [24, 25] which are conventionally tackled with distinct strategies.

To address the above challenges, we present OVER-NAV, a novel framework that strives to go over and beyond the current arts of IVLN solutions. To combat with the lack of explicit supervisions, we propose to incorporate Large Language Models (LLMs) and Open-Vocabulary Detection (OVD) to extract key information from the unattended data flow. Specifically, while LLMs are leveraged to identify keywords from the language instructions, the OVD detectors are employed to build the correspondence between the keywords and the visual observations. Such a strategy is ca-

pable of providing distilled supervision which is critical to understanding the unordered navigation data. As the OVD detector can scale up to novel categories, OVER-NAV is able to generalize to unseen scenes on the fly without the need of extra annotation and re-training, offering great flexibility over the methods using closed-set detectors [3, 25].

To better harness the extracted supervisory signals, we further introduce a structured representation, coded *Omnigraph*, to integrate multi-modal information along the tour. By leveraging a novel fusion mechanism, omnigraph can efficiently collect the pertinent knowledge for a more accurate navigation action. Moreover, the omnigraph representation can seamlessly support both discrete and continuous environments under a unified framework. We extensively evaluate OVER-NAV on a number of challenging benchmarks. The experimental results demonstrate the superiority of our method over the state-of-the-art approaches. In summary, our contributions are:

- A novel framework dubbed OVER-NAV, that, for the first time, incorporates LLMs and OVD into the IVLN paradigms to distill reliable and generalizable supervision signals from the unordered navigation data.
- A structured and general representation called Omnigraph that facilitates the utilization of multi-modal knowledge and can be generalized to different VLN settings.
- Superior performance on the IVLN tasks in both discrete and continuous environments.

2. Related Works

Vision-and-Language Navigation Vision-and-Language Navigation (VLN) [1, 10, 11, 15, 16, 21, 26, 28, 30, 36, 41] requires an agent with the ability to navigate a never-before-seen environment following a natural language instruction that describes the ground truth navigation path. There are two major settings in VLN benchmarks, discrete [1, 26, 36] and continuous environments [24, 40]. In the discrete setting, the VLN agent is limited to changing position and orientation by discrete amounts or predefined options, while the continuous setting provides a continuous range for the agent’s action. Iterative VLN [25] evaluates the agent in persistent environments, and the agent needs to utilize prior experience in the environment for better performance.

Persistent Environment and Iterative VLN The increasing amount and improved quality of 3D scene datasets [4, 39], and the high-performance navigation environment simulation platform [23, 31, 40, 44] significantly promote the development of navigation tasks, and make it possible to study the long-horizon tasks in persistent environments such as visual navigation [42, 43], multi-object navigation [42], visual room rearrangement [43], courier task [33], multi-target embodied QA [45], scene exploration and object search [9]. Iterative VLN [25] further studies VLN in

persistent environments and enriches the long-horizon visual navigation problem with natural language and linguistic information. Previous IVLN studies demonstrate that structured memory [3, 5, 7, 12, 14, 27, 35, 38, 42] is essential for IVLN agents. TourHAMT [25] tries to mitigate the problem in discrete environments by adding the history embedding from previous episodes to the current episode but fails to improve the performance. MAP-CMA [25] constructs semantic and occupancy maps [3] from the point cloud built by fine-tuned RedNet [19] and depth images for action prediction. However, point cloud construction requires depth sensors, and the RedNet is close-vocabulary and only limited to thirteen kinds of objects. This requires extra data collection labor for fine-tuning and prevents the agent from scaling to unseen environments and complicated concepts. TourHAMT/MAP-CMA can only be applied to discrete/continuous environments. Despite MAP-CMA’s success, it is hard to transfer its solution to discrete environments as semantic maps require continuous observations.

Open-Vocabulary Detection Open-vocabulary detection (OVD) aims to train object detectors beyond recognizing only base categories present in training labels and expand the vocabulary to detect novel categories. With the development of powerful language encoders [22] and contrastive image-text training [18, 37, 47], recent works transfer the language capabilities of these models to OVD [13, 20, 29, 46, 48, 49]. ViLD [13] distills the knowledge from a pretrained open-vocabulary image classification model into a two-stage detector following a teacher-student training paradigm. MDETR [20] and GLIP [29] take a single text query for the image then formulate detection as the phrase grounding problem. Owl-ViT [32] combines Vision Transformer [8], contrastive image-text pre-training [37] and end-to-end detection fine-tuning. We study OVD in persistent environments thus promoting the development of VLN.

3. OVER-NAV

The framework of our proposed method is depicted in Fig 2. Large Language Models (LLMs) extract keywords from the instruction of each episode. As the agent traverses the environment, it sends the keywords and observations along its path to the OVD detector for detection. The detection results are subsequently stored in the omnigraph, which is established and maintained to memorize the observed portion of the current environment. The omnigraph generates the keyword input for the agent’s action prediction. This input encapsulates the information of an ego-centric map, thereby aiding the prediction process.

3.1. OVD-based Omnigraph Construction

The OVER-NAV aims to construct an organized omnigraph that allows the IVLN agent to memorize and utilize the his-

tory information from previous episodes in the same tour. It incorporates three distinct processes: keyword extraction, keyword-panoramic detection, and omnigraph construction.

Keyword Extraction The IVLN agent receives an instruction each episode in the tour. A typical navigation instruction can be decomposed into two parts: milestones and actions. The milestones (hereinafter referred to as keywords) are related to the scene where the navigation task is issued. Conversely, actions are likely episodic and related to the agent’s current orientation. For instance, consider the instruction “*Head past the dining table and turn left towards the kitchen*”, the terms *dining table* and *kitchen* serve as milestones, enabling the agent to discern different directions and movements (e.g., *head past*, *turn left*) and verify the accuracy of previous actions. Thus the keywords provide the agent with a condensed understanding of the scene.

We propose employing Large Language Models (LLMs) [2, 34] for keyword extraction. The system prompt of GPT is provided in supplementary material. The GPT is fed instructions following the format defined in the prompt. Subsequently, the LLM responds to the query with the appropriately formatted keywords. One of the key benefits of using LLMs for keyword extraction lies in their flexibility because the keywords in instructions can vary greatly in length and category. For instance, keywords like “*counter with the blue top*” carry significant attributes for the agent’s reference, and LLMs can effectively identify such keywords with their strong in-context learning ability.

Keyword-Panoramic Detection After keyword extraction, the agent extracts a keyword set $K_I = \{k_i\}_{i=1}^N$ with N keywords from each instruction I . Throughout the navigation process of instruction I , the agent constantly observes the environment and captures images from its current position and orientation. Following the common practice of some VLN models, e.g., HAMT [6], we assume that the agent can acquire panoramic photos P_{pano} of the surrounding environment. This can be achieved by equipping the agent with a panoramic camera or rotating the agent 360° .

Upon reaching a position and capturing panoramic images, the agent can perform open-vocabulary detection using the keyword set K_I and the image P_{pano} . This position is subsequently recorded as a **viewpoint** in the agent’s memory. The OVD detector D generates a set of detection boxes $\{B_i, l_i, c_i\}_{i=1}^M$ as detection results, where each box B_i is with a label l_i and a confidence score c_i . Each box B_i contains four values $(x_{min}, y_{min}, x_{max}, y_{max})$ to locate the object in the panoramic image. These values can be used to calculate the relative heading of the detected object concerning the agent’s orientation. This can then be transformed into the absolute heading h_i in the coordinate system of the current environment by deducting the agent’s heading. Thus, the detection results stored in the agent’s memory for the current viewpoint are $\{B_i, l_i, c_i, h_i\}_{i=1}^M$.

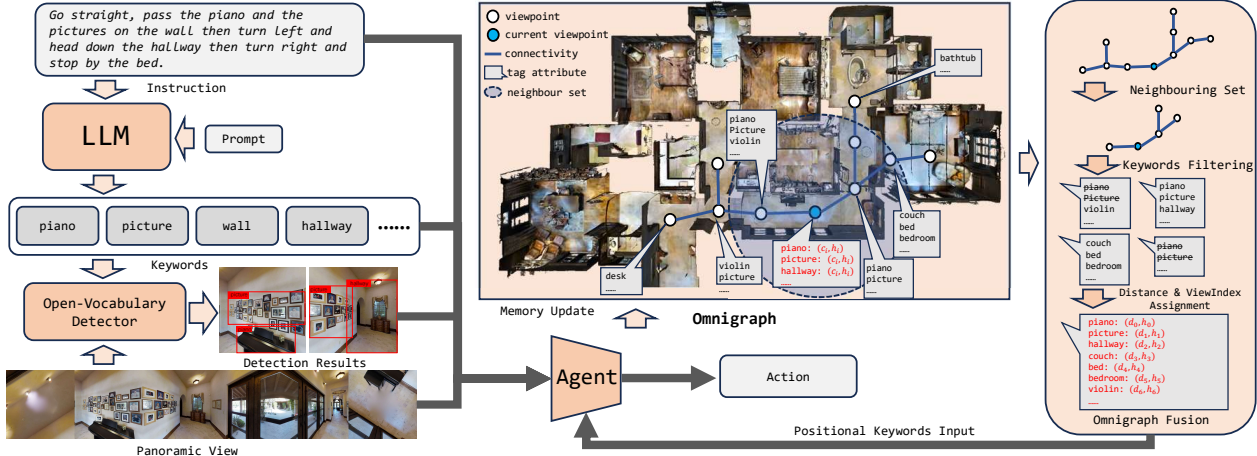


Figure 2. The overview of our proposed method. The instruction is sent to LLMs with the prompt to obtain keywords. The open-vocabulary detector receives the keywords and the panoramic view at the current position, and sends the detection results to the agent. With the detection results containing the distribution of detected objects, e.g., heading and confidence, the agent maintains the omnigraph that stores the information of visited viewpoints in previous episodes. Each viewpoint is tagged with keywords and distribution information. For inference, the omnigraph first collects the neighboring viewpoints and filters their keywords, then fuses the keywords with corresponding positional information, e.g., heading and confidence. Finally, the resulting positional keyword inputs are sent to the agent for prediction.

In each episode, the agent performs the detection with K_I and P_{pano} at every viewpoint along the navigation path. Each viewpoint along the path is tagged with detection boxes, labels, confidence, and absolute heading, which summarize the properties of viewpoints in the scene. We will further elaborate the details, e.g., the location of viewpoints, in Sec. 3.2.1 and Sec. 3.2.2.

Omnigraph Construction After keyword extraction and keyword-panoramic detection, all visited viewpoints are tagged with keywords. We then construct the **omnigraph** as a graph whose nodes are the viewpoints with keywords and edges are the connectivity between viewpoints. The omnigraph organizes the viewpoint, delineates the structure of the current scene, and outlines the distribution of various objects identified through panoramic keyword detection. It can be readily applied to both discrete and continuous environments. Furthermore, the open-vocabulary omnigraph can handle a wide array of keywords, and also empowers the model to discern the intrinsic relationship between keywords. For example, if the agent determines the relationship between the *living room* and the *television*, it can make an informed decision when instructed to go to the *living room*, even if only the *television* is present in the omnigraph.

Specifically, when the agent arrives at a new viewpoint a from a previous viewpoint b , and obtains the detection results, it incorporates the viewpoint a into the graph as a new node, and adds the undirected edge $\langle a, b \rangle$ to the omnigraph. If a is the starting point, only a is added to the graph. During the navigation, the omnigraph is incrementally refined in two ways: i) the discovery of viewpoints updates the nodes

and edges, ii) when the agent arrives at a previously visited viewpoint discovered in an earlier episode E_b in a new episode E_a , the keyword-panoramic detection will generate fresh detection results with the new instruction I_{E_a} , which might have different keywords from the previous instruction I_{E_b} . The new detection results are then employed to update the keywords associated with this viewpoint.

3.2. Omnigraph Fusion

After the omnigraph construction, the agent needs to exploit the information in the omnigraph at each step during its navigation. However, directly feeding the entire omnigraph to the agent is computationally intensive and inefficient as the agent only navigates a small area of the entire scene and the number of detection boxes might be overwhelmingly large. Hence, we fuse the information of the omnigraph within a local and ego-centric subgraph and send it to the agent. We reuse the agent’s instruction encoder to extract the embeddings of keywords in omnigraph subgraph, which not only eliminates the cost of the extra module but also preserves the semantic consistency between instruction embeddings and keyword embeddings. Then the extra information for each keyword is fused to enrich the keyword embeddings. Here we discuss the omnigraph fusion in discrete and continuous environments respectively. The overviews of the two agents are provided in supplementary materials.

3.2.1 OVER-NAV for Discrete Environments

In discrete VLN, the environment is represented as a set of pre-defined viewpoints, and the agent can navigate through

the connections between these viewpoints. Following previous studies, we integrate OVER-NAV into the History Aware Multimodal Transformer (HAMT) [6] framework to work with discrete environments.

HAMT [6] trains the agent with both text-modal and vision-modal inputs. HAMT agent encodes the input instruction $\mathcal{W} = (w_1, w_2, \dots, w_L)$ with L words as instruction embeddings $X = (x_{CLS}, x_1, x_2, \dots, x_L)$ using BERT, and encodes the panoramic observation $\mathcal{O}_t = (v_1^o, v_2^o, \dots, v_K^o)$ at step t with K different views as observation embeddings $O_t = (o_1, o_2, \dots, o_K, o_{stop})$. HAMT proposes hierarchical history encoding to encode the navigation history \mathcal{H}_t at step t , which consists of all the past panoramic observations $\mathcal{O}_{1, \dots, t-1}$ and performed actions $a_{1, \dots, t-1}$ before step t . The hierarchical history encoding produces history embedding $H_t = (h_{CLS}, h_1, \dots, h_{t-1})$. Then the history embedding and observation embedding are concatenated as vision modality, while instruction embedding serves as text modality. The cross-modal transformer fuses them and outputs the probability distribution of selecting different views of observation O_t as the predicted action.

Since the discrete environment is already discretized into viewpoints, the OVER-NAV agent performs object detection at each new viewpoint encountered, *i.e.*, at every step. For an agent positioned at viewpoint p , we prepare the omnigraph input using the following steps:

1. Neighbours Identification. Given distance d_n , we collect the neighboring viewpoints that are reachable from the current position within d_n steps as the neighbor set.
2. Inner-viewpoint Detection Box Filtering. For each neighbor, we filter out most of the detection boxes so that each keyword detected in this neighbor is associated with only one detection box with the highest detection confidence c_i .
3. Distance & View Index Assignment. For keyword k of neighbouring viewpoint v , we assign the distance d_v^k between v and the current position to k , and the view index h_v^k which is the direction of the next move the agent should take to go to v from the current position to k . Please note that both d_v^k and h_v^k are discrete rather than continuous, which makes the next step possible.
4. Cross-viewpoint Keyword Filtering. For those keywords that appear in more than one neighbour and thus have multiple d_v^k and h_v^k , we select the most frequent d_v^k and h_v^k as final attributes for k .

After these procedures, the agent obtains a set of distinct keywords and each keyword has a distance d_v^k and a view index h_v^k . Following HAMT, we use BERT to extract the embeddings of keywords, then use the [CLS] token embeddings E_{CLS} as the representation of each keyword. The view index h_v^k is converted to 4-dimension embeddings $E_{h_v^k} = (\sin \theta, \cos \theta, \sin \phi, \cos \phi)$ where θ and ϕ are the heading and elevation of view index h_v . Therefore,

the fused keyword embedding is computed as:

$$E_k = LN(W_{CLS}E_{CLS}) + LN(W_{h_v^k}E_{h_v^k}) + E_{d_v^k}, \quad (1)$$

where $E_{d_v^k}$ is the distance embedding, W_{CLS} and $W_{h_v^k}$ are learnable weights, and LN is layer normalization. Then all embeddings of keywords are concatenated:

$$E_{map} = \text{Concat}(E_{k_1}, E_{k_2}, \dots, E_{k_N}), \quad (2)$$

where N is the number of keywords and E_{k_i} is the fused keyword embedding of the i -th closest keyword k_i to the agent's current position.

The embedding E_{map} can be viewed as the text-modal context that indicates the position of every detected object in the memory. Therefore, we concatenate E_{map} to instruction embeddings X and send them to the agent as text modality.

3.2.2 OVER-NAV for Continuous Environments

VLN in continuous environments discards the pre-defined viewpoints for navigation and situates the agent in continuous environments with low-level actions. Following previous studies, we apply our OVER-NAV to continuous environments by incorporating OVER-NAV into a variant of Cross-Modal Attention [24], MAP-CMA [25].

MAP-CMA [25] is based on CMA [24], a common baseline in recent works. CMA is an end-to-end recurrent model that predicts actions from RGBD observations, the instruction and previous actions. CMA uses two recurrent networks, one for visual history tracking, and the other for instruction and visual features. MAP-CMA replace the RGBD observations with occupancy maps and semantic maps, which are built by using inverse pinhole camera projection model to 3D pointclouds. MAP-CMA use a Red-Net [19] feature encoder to form a semantic pointcloud, which is fine-tuned on thirteen common labels.

The application of OVER-NAV to MAP-CMA is similar to HAMT, except that OVER-NAV needs to decide the position of viewpoints, *i.e.*, where the keyword-panoramic detection should be performed. For an agent at position p , we prepare the omnigraph input with the following steps:

1. Viewpoint Discovery. Given discovery threshold d_{vp} as a hyper-parameter, when the agent arrives at a position whose distances to all recorded viewpoints in the memory are larger than d_{vp} , this position is registered as a new viewpoint v and the agent stores the observations P_{pano}^v . Different from Discrete VLN, we perform the keyword-panoramic detection with the stored observations P_{pano}^v at viewpoint v and keywords from current instruction when the agent arrives at a position whose distance to v is less than the hyper-parameter viewpoint detection threshold d_{det} . $d_{det} < d_{vp}$.
2. Neighbours Identification. Given a hyper-parameter d_n , we collect the neighboring viewpoints whose distance to the current position is less than d_n .

#	Model	PH	TH	PHI	IW	Val-Seen						Val-Unseen							
						TL	NE ↓	OS ↑	nDTW ↑	SR ↑	SPL ↑	t-nDTW ↑	TL	NE ↓	OS ↑	nDTW ↑	SR ↑	SPL ↑	t-nDTW ↑
1	HAMT					10.1 ±0.1	4.2 ±0.1	70 ±1	71 ±1	63 ±1	61 ±1	58 ±1	9.4 ±0.1	4.7 ±0.0	64 ±1	66 ±0	56 ±0	54 ±0	50 ±0
2	TourHAMT	✓	✓	✓	✓	9.4 ±0.4	5.8 ±0.1	56 ±1	59 ±0	45 ±1	43 ±1	45 ±0	10.0 ±0.2	6.2 ±0.1	52 ±2	52 ±0	39 ±1	36 ±0	32 ±1
3		✓	✓	✓		10.5 ±0.3	6.0 ±0.2	60 ±1	58 ±1	45 ±2	43 ±2	42 ±1	10.9 ±0.2	6.8 ±0.2	54 ±1	51 ±1	38 ±1	34 ±1	31 ±1
4		✓	✓			10.6 ±0.3	6.0 ±0.1	61 ±1	58 ±1	45 ±1	42 ±1	42 ±1	10.3 ±0.3	6.7 ±0.2	52 ±1	50 ±1	38 ±1	34 ±1	29 ±1
5		✓				10.9 ±0.3	6.1 ±0.1	60 ±2	58 ±1	45 ±1	42 ±1	41 ±0	11.0 ±0.6	6.7 ±0.1	52 ±2	51 ±0	38 ±0	34 ±0	28 ±1
6	Ours					9.9 ±0.1	3.7 ±0.1	70 ±0	73 ±1	65 ±1	63 ±1	62 ±0	9.4 ±0.1	4.1 ±0.1	66 ±1	69 ±0	60 ±1	57 ±0	55 ±1

Table 1. The comparison between our method, HAMT and TourHAMT on IR2R. PH: previous episodes’ history; TH: trainable history encoder; PHI: previous history identifier; IW: inflection weighting. TourHAMT fails to outperform HAMT, while our method achieves significant improvements in both val-seen and val-unseen datasets. We run each experiment 3 times and report metrics as $\bar{x} \pm \sigma_{\bar{x}}$.

#	Model	Val-Seen						Val-Unseen							
		TL	NE ↓	OS ↑	nDTW ↑	SR ↑	SPL ↑	t-nDTW ↑	TL	NE ↓	OS ↑	nDTW ↑	SR ↑	SPL ↑	t-nDTW ↑
1	CMA	7.8 ±0.4	8.8 ±0.6	27 ±3	42 ±3	18 ±3	17 ±3	39 ±1	7.5 ±0.3	8.8 ±0.2	26 ±1	44 ±1	19 ±1	18 ±1	38 ±2
2	TourCMA	8.0 ±0.4	8.2 ±0.9	30 ±2	44 ±2	20 ±3	19 ±2	40 ±1	7.8 ±0.1	9.0 ±0.2	26 ±1	42 ±1	18 ±0	17 ±1	36 ±1
3	PoolCMA	7.2 ±0.5	9.1 ±0.4	24 ±4	41 ±2	17 ±4	16 ±2	37 ±2	7.3 ±0.2	9.0 ±0.3	23 ±1	42 ±1	16 ±1	15 ±0	36 ±2
4	PoolEndCMA	7.6 ±0.8	8.9 ±0.9	27 ±3	42 ±3	18 ±4	17 ±2	38 ±2	6.9 ±0.2	8.7 ±0.2	25 ±2	44 ±1	18 ±1	16 ±1	38 ±2
5	MAP-CMA	9.4	6.4	48	56	39	36	52	8.5	6.8	44	54	35	32	47
6	Ours	9.5 ±0.9	5.8 ±0.9	49 ±4	59 ±2	39 ±2	36 ±2	56 ±2	8.8 ±0.6	6.5 ±0.2	45 ±2	56 ±1	35 ±1	33 ±1	50 ±2

Table 2. The performance of our method on IR2R-CE. The experiment results of all previous methods are copied from [25]. Our method and MAP-CMA both use inferred semantics and iterative map construction. Our method gains 4% and 3% t-nDTW improvement on val-seen and val-unseen datasets respectively. We run each experiment 3 times and report metrics as $\bar{x} \pm \sigma_{\bar{x}}$.

3. Inner-viewpoint Detection Box Filtering. This part is the same as Sec. 3.2.1.
4. Distance & Heading Assignment. Similar to Sec. 3.2.1, we assign heading h_v^k and distance d_v^k to each keyword. In this case, d_v^k and h_v^k are continuous.
5. Cross-viewpoint Keyword Filtering. We select the detection box with the highest score for each keyword.

Similar to Sec. 3.2.1, the agent receives a set of distinct keywords and each keyword has a heading h_v^k and distance d_v^k . We use the instruction encoder of MAP-CMA to extract the embedding of every keyword. Then we fuse the keyword with heading and distance embeddings using linear transformation. Finally, we perform the attention operation with the instruction embedding as query and the fused keyword embedding as key/value. The operation result of the attention is sent to the model for action prediction.

4. Experiments

Following previous studies [25], we present the experiment results on IR2R and IR2R-CE in this section.

Dataset IR2R [25] is an iterative version of R2R [1] dataset, a common evaluation benchmark for Vision-and-Language Navigation in discrete environments collected from Matterport3D [4]. Same as R2R, IR2R is split into training/val-seen/val-unseen set. IR2R training set contains 14025 episodes from 61 scenes, and each scene has 3 tours. IR2R val-seen and val-unseen set has 53/11 scenes, 1011/2349 episodes, and 159/33 tours respectively. IR2R-CE [25] is the iterative version of R2R-CE [24] benchmark, which

transforms the instruction and path in R2R to continuous form and uses Habitat [40] for environment simulation.

Evaluation Metric Following previous studies [25], we use t-nDTW [25] to evaluate the performance of different methods in IVLN. t-nDTW is the tour-version of normalized dynamic time warping (nDTW) [17], which is a common metric for evaluation in VLN by measuring the normalized similarity between the ground truth path and the agent’s navigation path. Please refer to more details in [25].

Implementation Details We implement our method on [25] and retain all the hyper-parameters of VLN agents, including HAMT and MAP-CMA. The LLM we use for keyword extraction is GPT-3.5 [2], and the prompts are provided in supplementary materials. For OVD detection, we use OWL-ViT [32] with ViT-L/14 backbone [8]. For discrete environments, we set neighbour distance d_n as 3. For continuous environments, we set neighbour distance d_n as 7, discovery threshold d_{vp} as 1, and detection threshold d_{det} as 0.25. All experiments are conducted on two RTX-3090 GPUs with 24GB memory. We train the model on one GPU and run the detection model on the other. Please refer to supplementary materials for details.

Experiment Result The experiment results on IR2R and IR2R-CE are shown in Table 1 and Table 2. In Table 1, we compare our method to HAMT baseline and four different variants of TourHAMT. The performance of TourHAMT is copied from [25]. For each model, we present Total Length (TL), Navigation Error (NE), Oracle Success (OS), nDTW, Success Rate (SR), and Success weighted by Path

#	Model	Val-Seen							Val-Unseen						
		TL	NE ↓	OS ↑	nDTW ↑	SR ↑	SPL ↑	t-nDTW ↑	TL	NE ↓	OS ↑	nDTW ↑	SR ↑	SPL ↑	t-nDTW ↑
1	Baseline	10.1 ±0.1	4.2 ±0.1	70 ±1	71 ±1	63 ±1	61 ±1	58 ±1	9.4 ±0.1	4.7 ±0.0	64 ±1	66 ±0	56 ±0	54 ±0	50 ±0
6	Type-I	10.1 ±0.2	3.8 ±0.1	70 ±1	73 ±1	65 ±1	62 ±1	61 ±1	9.7 ±0.3	4.2 ±0.1	66 ±1	68 ±0	59 ±1	56 ±0	52 ±1
6	Type-II	10.2 ±0.3	3.7 ±0.1	71 ±1	73 ±1	65 ±1	62 ±1	62 ±0	9.8 ±0.2	4.2 ±0.2	66 ±1	68 ±1	59 ±1	56 ±0	53 ±1
6	Ours	9.9 ±0.1	3.7 ±0.1	70 ±0	73 ±1	65 ±1	63 ±1	62 ±0	9.4 ±0.1	4.1 ±0.1	66 ±1	69 ±0	60 ±1	57 ±0	55 ±1
6	Type-III	9.9 ±0.2	3.6 ±0.1	70 ±0	74 ±1	65 ±0	63 ±1	63 ±0	9.4 ±0.2	4.0 ±0.0	68 ±1	69 ±1	61 ±1	58 ±1	56 ±1

Table 3. The ablation of open-vocabulary keywords. Type-I, Type-II and Type-III adopt different keyword strategies compared to our method. Type-I limits the keywords to 12 categories and obtains minimal improvement among these models. Type-II retains the attributes of the 12 keywords and slightly improves the performance of Type-I. Ours retains all open-vocabulary keywords and further gains 2%. Type-III’s memory contains the detection results in all viewpoints for the keywords of the current instruction in advance, thus achieving the best performance.

#	Model	Val-Seen		Val-Unseen	
		nDTW ↑	t-nDTW ↑	nDTW ↑	t-nDTW ↑
1	Ours ($d_n=1$)	73.0	62.1	67.5	53.4
2	Ours ($d_n=2$)	73.3	62.6	68.1	54.0
3	Ours ($d_n=3$)	73.0	62.3	68.7	54.9
4	Ours ($d_n=4$)	72.1	61.2	68.1	54.4
5	Ours ($d_n=5$)	71.9	61.2	68.0	54.2

Table 4. The performance of our method on IR2R with different neighbour distances d_n , which determines the size of neighbouring set. Our method achieves the best performance with $d_n = 3$.

#	Model	DISTANCE	VIEW INDEX	Val-Seen		Val-Unseen	
				nDTW ↑	t-nDTW ↑	nDTW ↑	t-nDTW ↑
1	Ours			70.7	58.6	66.1	50.6
2	Ours	✓		72.2	61.3	67.2	51.3
3	Ours		✓	72.5	62.0	68.2	53.4
4	Ours	✓	✓	73.0	62.3	68.7	54.9

Table 5. The ablation of omnigraph information. We remove the distance d_v and viewindex h_v for all keywords. Both distance d_v and viewindex h_v contribute to the performance improvement, which indicates the importance of positional information and structured memory.

Length (SPL) besides t-nDTW. These six metrics are calculated on the episode level. t-nDTW is the tour-based metric for IVLN evaluation. As shown in Table 1, TourHAMT fails to improve the performance using tour navigation history, while our method gains 4% and 5% improvement in val-seen and val-unseen set respectively. Moreover, our method achieves better performance in episode-level metrics, including SPL, nDTW, SR, OS and NE. Note that our method is based on HAMT, the superior performance demonstrate the benefit of utilizing the history of previous episodes.

Table 2 shows the performance in continuous environments. We compare our method to CMA [24] and its three variants, TourCMA, PoolCMA and PoolEndCMA [25]. We also compare our method to MAP-CMA, a strong baseline for IVLN-CE. Our method gains 4% and 3% t-nDTW improvement in val-seen and val-unseen set respectively, while achieving better or comparable performance in

episode-level metrics. Note that the performance of MAP-CMA [25] is copied from the original paper. Our method is implemented on MAP-CMA with the same experiment setting, i.e., inferred semantics and iterative map construction, on which MAP-CMA achieves its best performance.

5. Ablation Study

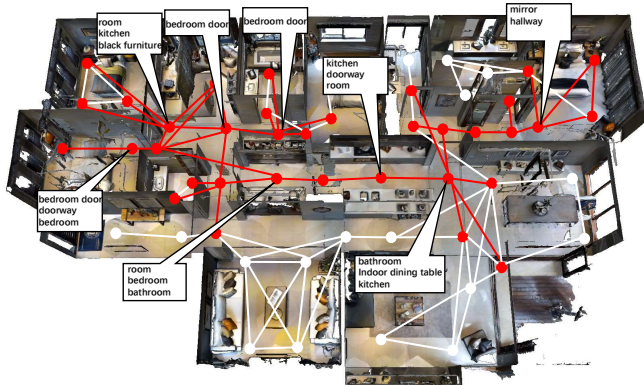
5.1. Ablation on Open-Vocabulary Keywords

To demonstrate the superiority of the open-vocabulary keywords in our method, here we ablate the Open-Vocabulary strategy in the three variants of our method in Table 3.

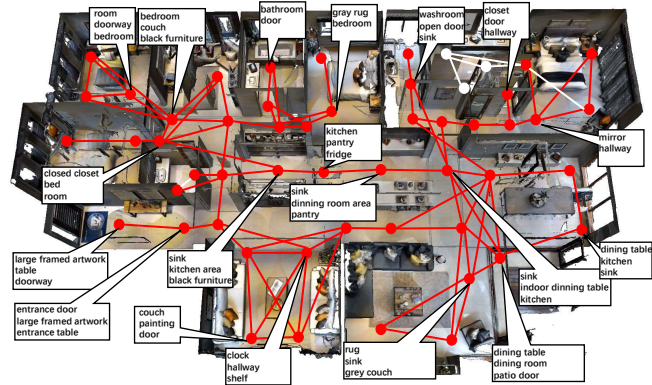
Type-I Following previous studies[3], we collect and focus on the 12 most common object categories in R2R environment: {*chair, table, cushion, cabinet, shelving, sink, dresser, plant, bed, sofa, counter, fireplace*} (sorted in descending order by number of object instances). Type-I discards the Keyword Extraction stage and performs the Keyword-Panoramic Detection with the above 12 categories as keywords. In this case, the OVD detector in our method detects the objects in the observations in a close-vocabulary way, i.e., all detection boxes sent to the agent’s memory are limited to the 12 categories above.

Type-II Compared to the original method, Type-II only sends the detection boxes with labels that contain at least one of the 12 most common objects above. For example, “*marble kitchen counter*” contains “*counter*” while “*kitchen island*” does not contain any class names above. Thus only “*marble kitchen counter*” will be sent to the detector and then to the agent’s memory. In contrast, Type-I only has “*counter*” in its keywords.

Type-III Type-III shows the performance of models with oracle keyword detections. After training the agent, we evaluate the agent twice on the validation seen/unseen dataset and do not clear the agent’s memory after the first evaluation. Thus the agent’s memory contains the detection result of the first evaluation which shares the same keywords to the second evaluation. Type-III simulates the scenarios where all the keywords in the evaluation have been detected and saved to the memory, while the parameters of



(a) Omnigraph visualization after 5 episodes in a 100-episode tour.



(b) Omnigraph visualization after 50 episodes in a 100-episode tour.

Figure 3. The visualization of Omnigraph in our method during a 100-episode tour. As the tour proceeds, the omnigraph becomes larger with more viewpoints and more connections. The keywords attached to viewpoints become more precise and diverse. We show 3 keywords for each viewpoint at most and omit the extra information (e.g., heading) for simplicity.

the agent are not updated.

Our method outperforms Type-I and Type-II models on both Val-Seen and Val-Unseen datasets. The difference between the four models indicates the performance contribution of open-vocabulary detection. Note that Type-III achieves the best performance because its memory contains the exact keywords of every episode in advance.

5.2. Ablation on Structured Memory

To demonstrate the effectiveness of our structured memory, we ablate the two important attributes for each keyword in omnigraph, distance d_v and view index h_v . The experiments are conducted in the discrete environment, i.e., IR2R.

Specifically, OVER-NAV fuses distance d_v and view index h_v with each keyword. d_v indicates the distance between the agent’s current position and the viewpoint where the keyword is detected, while h_v represents the direction the agent should move towards to go to the viewpoint where the keyword is detected. Table 5 shows the ablation experiment results. It can be observed that both attributes, distance d_v and view index h_v contribute to the performance improvement. The ablation of each attribute will decrease the performance, and the view index h_v is more important than distance d_v , because view index h_v indicates which direction the agent should move towards more clearly.

5.3. Sensitivity for Hyper-Parameter Depth d_n

Hyper-parameter depth d_n is the distance threshold for neighbour identification as described in Section 3. Neighbour identification only collects the viewpoints whose distances to the agent’s current position are less than threshold d_n , and sends the collected viewpoints to the following procedures. Here we analyze the sensitivity for d_n . The performance of our method with different d_n on IR2R is shown in Table 4. The best performance is achieved when $d_n = 3$.

5.4. Visualization

We visualize the omnigraph in Fig. 3 on IR2R. The tour shown in Fig. 3 contains 100 episodes in the same scene, and Fig. 3a and Fig. 3b shows the omnigraph after 5 episodes and 50 episodes respectively. The dots/lines indicate the viewpoint and connectivity between them respectively. The red dots/lines are the viewpoints/connections visited by the agent in previous episodes, while the white dots/lines are unvisited. We show 3 keywords for some viewpoints at most and omit the extra information (e.g., heading) for simplicity. As shown in Fig. 3a, the agent navigates a large portion of the scene (66% of all viewpoints) with a small number of episodes (5% of the instructions). The keywords from the 5 instructions are limited, but we can still observe that some open-vocabulary keywords are correctly detected, e.g., black furniture and indoor dining table. In Fig. 3b, most of the viewpoints are recorded in the omnigraph as well as the connectivity. The keywords attached to the viewpoints are more diverse and precise, which enhances the representability of the omnigraph and provides more accurate information for the agent.

6. Conclusion

We propose an open-vocabulary-based method, OVER-NAV for Iterative Vision-Language Navigation. OVER-NAV incorporates LLMs and an Open-Vocabulary detector to construct an omnigraph, which consists of viewpoints and connections with keywords to describe the distribution of key objects that are important for the agent’s navigation. Extensive experiments in both discrete and continuous environments demonstrate that omnigraph is a superior and more general structured memory to memorize and describe the navigation scene, enabling the agent to utilize the navigation history of previous episodes in IVLN for better performance.

References

- [1] 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. 1, 2, 6
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 3, 6
- [3] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semantic maps and representations from egocentric views. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 964–972, 2021. 2, 3, 7
- [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 Computer Society, 2017. 2, 6
- [5] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations*, 2020. 3
- [6] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847, 2021. 3, 5
- [7] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *International Conference on Learning Representations*, 2019. 3
- [8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 3, 6
- [9] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 538–547, 2019. 2
- [10] 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. *Advances in Neural Information Processing Systems*, 31, 2018. 2
- [11] Chen Gao, Xingyu Peng, Mi Yan, He Wang, Lirong Yang, Haibing Ren, Hongsheng Li, and Si Liu. Adaptive zone-aware hierarchical planner for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14911–14920, 2023. 2
- [12] Georgios Georgakis, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, and Kostas Daniilidis. Learning to map for active semantic goal navigation. In *International Conference on Learning Representations*, 2022. 3
- [13] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations*, 2022. 3
- [14] Joao F Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8476–8484, 2018. 3
- [15] Jingyang Huo, Qiang Sun, Boyan Jiang, Haitao Lin, and Yanwei Fu. Geovln: Learning geometry-enhanced visual representation with slot attention for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23212–23221, 2023. 2
- [16] Minyoung Hwang, Jaeyeon Jeong, Minsoo Kim, Yoonseon Oh, and Songhwai Oh. Meta-explore: Exploratory hierarchical vision-and-language navigation using scene object spectrum grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6683–6693, 2023. 2
- [17] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *NeurIPS Visually Grounded Interaction and Language (ViGIL) Workshop*, 2019. 6
- [18] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021. 3
- [19] Jindong Jiang, Lunan Zheng, Fei Luo, and Zhijun Zhang. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation. *arXiv preprint arXiv:1806.01054*, 2018. 3, 5
- [20] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021. 3
- [21] Aishwarya Kamath, Peter Anderson, Su Wang, Jing Yu Koh, Alexander Ku, Austin Waters, Yinfei Yang, Jason Baldridge, and Zarana Parekh. A new path: Scaling vision-and-language navigation with synthetic instructions and imitation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10813–10823, 2023. 2
- [22] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019. 3

- [23] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. [2](#)
- [24] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer, 2020. [2](#), [5](#), [6](#), [7](#)
- [25] Jacob Krantz, Shurjo Banerjee, Wang Zhu, Jason Corso, Peter Anderson, Stefan Lee, and Jesse Thomason. Iterative vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14921–14930, 2023. [1](#), [2](#), [3](#), [5](#), [6](#), [7](#)
- [26] 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](#)
- [27] Obin Kwon, Jeongho Park, and Songhwai Oh. Renderable neural radiance map for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9099–9108, 2023. [3](#)
- [28] Jialu Li and Mohit Bansal. Improving vision-and-language navigation by generating future-view image semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10803–10812, 2023. [2](#)
- [29] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022. [3](#)
- [30] Xiangyang Li, Zihan Wang, Jiahao Yang, Yaowei Wang, and Shuqiang Jiang. Kerm: Knowledge enhanced reasoning for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2583–2592, 2023. [2](#)
- [31] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. [2](#)
- [32] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection with vision transformers. In *European Conference on Computer Vision*, pages 728–755. Springer, 2022. [3](#), [6](#)
- [33] Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Andrew Zisserman, Raia Hadsell, et al. Learning to navigate in cities without a map. *Advances in neural information processing systems*, 31, 2018. [2](#)
- [34] OpenAI. Gpt-4 technical report, 2023. [3](#)
- [35] Benjamin Planche, Xuejian Rong, Ziyang Wu, Srikrishna Karanam, Harald Kosch, YingLi Tian, Jan Ernst, and Andreas Hutter. Incremental scene synthesis. *Advances in Neural Information Processing Systems*, 32, 2019. [3](#)
- [36] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020. [2](#)
- [37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. [3](#)
- [38] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 400–418. Springer, 2020. [3](#)
- [39] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. [2](#)
- [40] 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](#), [6](#)
- [41] 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/CVF conference on computer vision and pattern recognition*, pages 6629–6638, 2019. [2](#)
- [42] Saim Wani, Shivansh Patel, Unnat Jain, Angel Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *Advances in Neural Information Processing Systems*, 33:9700–9712, 2020. [2](#), [3](#)
- [43] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5922–5931, 2021. [2](#)
- [44] 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](#)
- [45] Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, Tamara L Berg, and Dhruv Batra. Multi-target embodied

- question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2019. 2
- [46] Alireza Zareian, Kevin Dela Rosa, Derek Hao Hu, and Shih-Fu Chang. Open-vocabulary object detection using captions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14393–14402, 2021. 3
- [47] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133, 2022. 3
- [48] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, et al. Regionclip: Region-based language-image pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16793–16803, 2022. 3
- [49] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *European Conference on Computer Vision*, pages 350–368. Springer, 2022. 3