Structured Scene Memory for Vision-Language Navigation

Hanqing Wang[1], Wenguan Wang[2*], Wei Liang[1*], Caiming Xiong[3], Jianbing Shen[1,4]

[1]Beijing Institute of Technology [2]ETH Zurich [3]Salesforce Research [4]Inception Institute of Artificial Intelligence

https://github.com/HanqingWangAI/SSM-VLN

Abstract

Recently, numerous algorithms have been developed to tackle the problem of vision-language navigation (VLN), i.e., entailing an agent to navigate 3D environments through following linguistic instructions. However, current VLN agents simply store their past experiences/observations as latent states in recurrent networks, failing to capture environment layouts and make long-term planning. To address these limitations, we propose a crucial architecture, called Structured Scene Memory (SSM). It is compartmentalized enough to accurately memorize the percepts during navigation. It also serves as a structured scene representation, which captures and disentangles visual and geometric cues in the environment. SSM has a collect-read controller that adaptively collects information for supporting current decision making and mimics iterative algorithms for long-range reasoning. As SSM provides a complete action space, i.e., all the navigable places on the map, a frontier-exploration based navigation decision making strategy is introduced to enable efficient and global planning. Experiment results on two VLN datasets (i.e., R2R and R4R) show that our method achieves state-of-the-art performance on several metrics.

1. Introduction

As a crucial step towards building intelligent embodied agents, autonomous navigation has long been studied in robotics. Since Anderson *et al.* [3] extended prior efforts[9, 37] in instruction based navigation into photo-realistic simulated scenes [6], vision-language navigation (VLN) has recently attracted increasing attention in computer vision community. Towards the goal of enabling an agent to execute navigation instructions in 3D environments, current representative VLN methods made great advances in: **i)** developing more powerful learning paradigms[50, 48]; **ii)** exploring extra supervision signals from synthesized data[13, 45, 14] or auxiliary tasks [48, 25, 33, 55]; **iii)** designing more efficient multi-modal embedding schemes[24, 40, 49]; and **iv)** making more intelligent path planning [28, 34, 47].
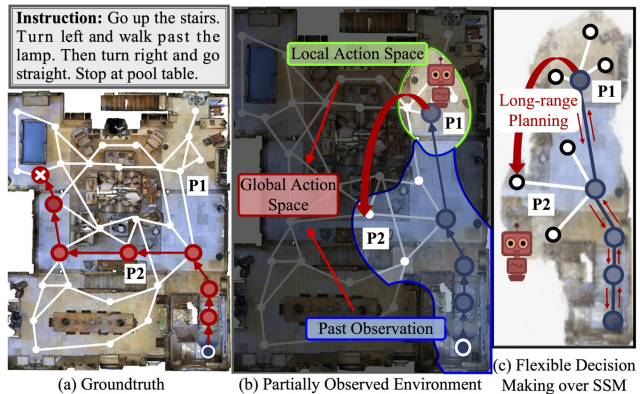

Figure 1: In the partially observed environment (b), our agent builds SSM (c) for persistent memorization and topological scene representation. SSM supports long-range planning over the global action space, thus our agent can easily change its navigation direction (*i.e.*, P1) by directly 'jumping' to any previously visited position (*e.g.*, P2), resulting in robust navigation.

Despite these progresses, current VLN models typically follow[3] to tackle the task through a sequence-to-sequence (Seq2Seq) framework, which maps instructions and online observations to navigation actions. With such a design, the perceived information are embedded and mixed in the internal recurrent units. This prohibits the agent from directly accessing to its past observations and understanding the environment layout. Taking Fig. 1 as an example. VLN agents execute instructions in a partially observed environment (Fig. 1(b)). As current VLN agents simply encode past perceptions (blue area in Fig. 1(b)) as hidden network states, their decision-making is only limited within a local action space (*i.e.*, currently navigable directions – green area in Fig. 1(b)). Thus they tend to make sub-optimal navigation decisions, failing to perform path planning over the explored environment (*i.e.*, global action space in Fig.1(b)).

In robotics, map-building is long studied for facilitating path planning [46, 15]. Inspired by classic simultaneous localization and mapping (SLAM) models [46, 19], recent efforts make use of deep learning techniques for constructing more efficient space representations [17, 12, 7, 8]. On the other hand, making path planning in VLN, in essence, requires efficiently modeling long-term dependencies within

*Corresponding author: *Wenguan Wang*.

online observation sequences. However, the states (*internal* memory) in recurrent networks are latent and shown inherently unstable over long timescales [44]. Alternatively, neural networks with *external* memories provide a feasible solution, which uses a compartmentalized memory to preserve and recall long-term information and has been shown successful in language modeling [44] and deep reinforcement learning (RL) agents in 3D environments [38, 39].

Built off these two lines of previous efforts, we propose an essential, outside memory architecture, called Structured Scene Memory (SSM). SSM online collects and precisely stores the percepts during navigation (see Fig. 1(b-c)). It is graph-structured, yielding a topological representation of the environment. Thus SSM is able to perform long-term memorization and capture the environment layouts, allowing our agent to make efficient planning. In addition, SSM delivers a global action space – all the navigable locations within the explored area, enabling flexible decision making. As shown in Fig.1(c), when necessary, our agent can simply move off its current navigation direction (*i.e.*, P1) and travel to another previously visited far location (*i.e.*, P2).

Specifically, SSM consists of nodes that embed visual information in explored locations and edges that represent geometric relations between connected locations. As both visual and geometric cues are captured and disentangled, instructions can be better grounded onto the visual world, *i.e.*, correlating perception and action related descriptions with nodes and edges in SSM respectively. SSM is equipped with a collect-read controller, which adaptively reads content from the memory, depending on current navigation context. The controller also mimics iterative algorithms for comprehensive information gathering and long-range reasoning. Hence, SSM brings a global action space but its gradually increased scale will make policy learning harder. We propose a frontier-exploration based decision making strategy that addresses this issue elegantly and efficiently.

Extensive experiments on Room-to-Room (R2R)[3] and Room-for-Room (R4R)[26] datasets demonstrate the effectiveness of the full approach and core model designs.

2. Related Work

Vision-Language Navigation (VLN). Since the release of R2R dataset [3], numerous VLN methods are proposed, which mainly focus on the use of deep multimodal learning to arrive at navigation policies. For instance, from imitation learning (IL) [3], to the hybrid of model-free and model-based RL [50], to the ensemble of IL and RL [48], different learning regimes are developed. For further boosting learning, some methods try to mine extra supervisory signals from synthesized samples [13, 45, 14] or auxiliary tasks [48, 25, 33, 55]. Hence, for more intelligent path planning, the abilities of self-correction [28, 34] and active exploration [47] are addressed. Some other recent studies also

address environment-agnostic representation learning [49], fine-grained instruction grounding [23, 40, 22], web image-text paired data based self-pretraining [35, 18], or perform VLN in continuous environments [29].

Existing VLN agents are built upon Seq2Seq models [3], where past observations are encoded as the hidden states of recurrent units. Thus they struggle to precise memorization over long time lags and fail to explore environment layouts. In contrast, our agent is empowered with an external, structured memory for effective space representation and efficient memory operation. It provides a global action space and allows flexible decision making. A concurrent work [11] also addresses planning over topological space representation. In [11], visual and geometric information of the environment are mixed together in the scene map. But these two are disentangled in ours, facilitating instruction grounding. Hence, to avoid a huge action space, we propose a frontier-exploration based decision making strategy, with a navigation utility driven policy learning regime. However, [11] requires a specifically designed planner with a subset of the decision space and trains the agent only with IL.

Neural Networks with External Memory. Many tasks require capturing structures within sequential data, such as machine translation and question answering. Though recurrent networks [21] have been shown Turing-Complete [43] and applicable for sequential learning, their ability in modeling long-term data structure is limited, as their memory are latent network states. Recent work [16] further explores augmenting networks with outside memory, which allows for long-term memorization as well as explicit memory manipulation (such as read and write operations). Some representative ones show remarkable success in language modeling [16, 51, 44, 4], robot control [38, 39], and meta-learning [41]. Inspired by their novel ideas, we develop a structured, external memory architecture, which can store percepts during long-time exploration and capture environment layouts. With a collect-read controller that addresses adaptive context information gathering, we believe our memory design is well-suited to the performance of the VLN task.

Space Representation and Path Planning in Navigation. To perform autonomous navigation, it is crucial to humanize a robot into learning spatial environments and making planning. Classic strategies typically build space representation via SLAM [46, 19], fusing multi-modal signals, such as LIDAR, depth, or structure from motion. With the estimated maps, path planning is made [27, 31, 5] for traveling to goal locations. Recent efforts focus on learning navigation policies over spatial representations [17, 39, 54], topological maps [42, 10, 52, 8, 7], or designing trainable planners [32, 11]. As for VLN, current methods rarely make use of scene maps, and thus cannot make robust planning. Although some ones address better planning through rule-based beam search [13], self-correction [28, 34], or active
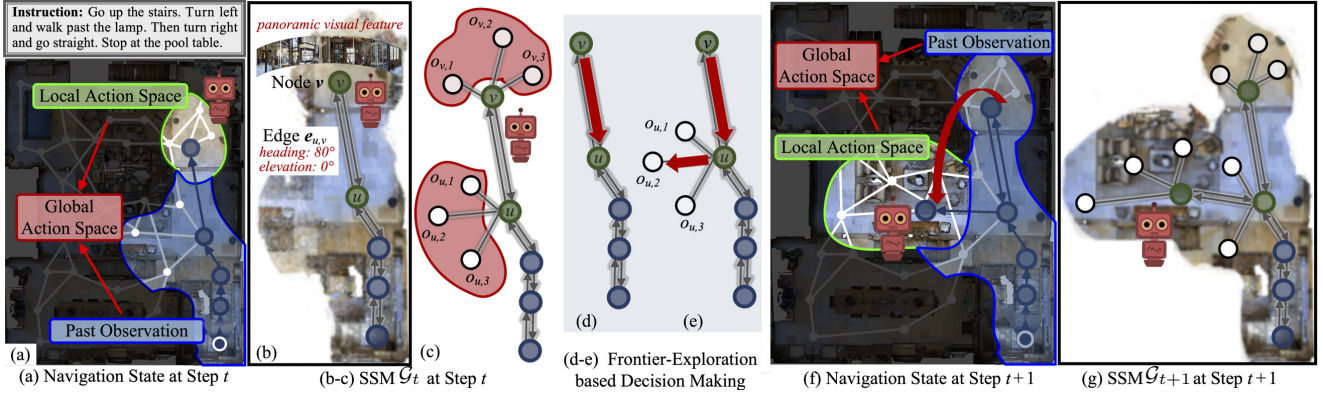
Figure 2: Previous VLN agents are limited to a local action space, while SSM allows our agent to store past observations, explicitly model visual and geometric cues in environmental layout, and make robust VLN planning over the global action space. Please see §3 for details.

perception [47], they still suffer from a local action space and latent memory. Instead, we explore and store scene representation through a structured memory, which provides a global action space and facilitates long-term reasoning. Hence, a frontier-exploration based decision making strategy is adopted to prevent the rapidly expanding action space from causing expensive learning and slow convergence.

3. Our Approach

Task Setup and Notations. Following the standard VLN setup [3], an embodied agent is required to online navigate through environments to reach target locations, according to language instructions. To be able to focus on high-level planning, in [3], the environment is assumed to be viewed from a set of sparse points and the navigability between the points is given. Formally, let us denote by $X = \{x_l\}_l$ the textual embedding of the instruction with $|X|$ words. At each step t, the agent observes a panoramic view [13], which is discretized into 36 single views (*i.e.*, RGB images). Hence, there are K_t navigable viewpoints $\{o_{t,k}\}_{k=1}^{K_t}$ that are reachable and visible. Each navigable viewpoint $o_{t,k}$ is associated with a visual feature $f_{t,k}$ and orientation feature $r_{t,k} = (\cos\theta_{t,k}, \sin\theta_{t,k}, \cos\phi_{t,k}, \sin\phi_{t,k})$, where θ and ϕ are the angles of heading and elevation, respectively.

Previous Solutions to VLN. Previous VLN agents [3, 45], in general, are built as a recurrent action selector based on received observations, instructions, and prior latent state. At each step t, the selector updates its internal state through:

$$h_t = \text{LSTM}([X, O_t], h_{t-1}), \qquad (1)$$

where $O_t = \{o_{t,k}\}_{k=1}^{K_t} = \{[f_{t,k}, r_{t,k}]\}_{k=1}^{K_t}$, and '$[\cdot]$' indicates the concatenation operation.

Current navigable viewpoints $\{o_{t,k}\}_{k=1}^{K_t}$ form the action space $\mathcal{A}_t$, from which the moving action a_t is selected:

$$a_t = \underset{k}{\arg\max}\; p(o_{t,k}). \qquad (2)$$

Here $p(o_{t,k}) = \text{softmax}_k(o_{t,k}^\top W_o^p h_t)$, where W_o^p is a learnable parameter matrix. Defined in this way, $p(o_{t,k})$ returns the probability of the viewpoint $o_{t,k}$ to be acted on.

Our Idea. With above brief descriptions of the core idea of existing VLN models, some limitations can be exemplified. First, they fail to explore environmental layouts. Second, their planning ability is limited to the latent memory h, which cannot precisely store and directly recall their past experience, *i.e.*, $\{O_1, \cdots, O_t\}$. Third, they suffer from a local action space $\mathcal{A}_t$, *i.e.*, immediate observation $\{o_{t,k}\}_{k=1}^{K_t}$. Thus their navigation behaviors tend to be reactive and previous wrong decisions are hard to be corrected.

To address these limitations, we extend existing *internal memory* based VLN agents with an *external, structured memory*, SSM. Concretely, SSM builds and stores scene layouts and allows easy access to past percepts. SSM also redefines the action space as the set of all the navigable places on the map, allowing making global planning. Next, we will first describe the way of constructing SSM (§3.1). Then we introduce iterative reasoning (§3.2) and frontier-exploration based two-step decision making (§3.3), which are two essential components of our SSM based navigator.

3.1. Structured Scene Memory (SSM) Construction

During navigation, our agent online constructs and maintains SSM as an episodic, topological map of its observed environment. SSM is derived from its past percepts and organized in the form of a graph. Specifically, at time step t, SSM is denoted as a directed graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$, where nodes $v \in \mathcal{V}_t$ correspond to previously visited places and edges $e_{u,v} = (u, v) \in \mathcal{E}_t$ to connections (see Fig. 2(a-b)).

Visual and Geometric Information Disentangled Layout Representation. Each node $v \in \mathcal{V}^t$ is associated with an embedding v to capture visual information at the position it stands for:

$$v = \{f_{v,k}\}_{k=1}^{K_v}. \qquad (3)$$

Here v encodes the visual features of the panorama $\{o_{v,k}\}_{k=1}^{K_v}$ at position v. For ease of notation, the symbols $f_{v,k}$ and K_v are slightly reused to represent the k-th single-view visual feature and total number of navigable viewpoints at location v, respectively.

For each edge $e_{u,v} \in \mathcal{E}_t$, its embedding $\boldsymbol{e}_{u,v}$ encodes geometric cues from position u to v:

$$\boldsymbol{e}_{u,v} = \boldsymbol{r}_{u,k_v} = (\cos\theta_{u,k_v}, \sin\theta_{u,k_v}, \cos\phi_{u,k_v}, \sin\phi_{u,k_v}). \quad (4)$$

Here $\boldsymbol{r}_{u,k_v}$ is slightly reused to represent the orientation feature of k_v-th viewpoint at position u, from which node v can be navigated. Note that $\boldsymbol{e}_{u,v} \neq \boldsymbol{e}_{v,u}$.

SSM constructs a topological scene representation from precisely memorized past observations. In addition, visual and geometric cues are disentangled in the node and edge embeddings, respectively. This allows better grounding instructions on the visual world (will be detailed in §3.2).

Global Action Space Modeling. Another distinct advantage of SSM is that it provides a global action space. In SSM, each node v is associated with the panoramic view at the corresponding location. Thus v can be divided into K_v sub-nodes $\{o_{v,k}\}_{k=1}^{K_v}$ (see Fig. 2(c)), and each sub-node represents a navigable viewpoint $o_{v,k}$ (*i.e.*, reachable, observable but unvisited). All the sub-nodes in $\mathcal{G}_t$ form the action space $\mathcal{A}_t = \{o_{v,k}\}_{v=1,k=1}^{|\mathcal{V}_t|,K_v}$ at step t. That means any navigable place on the map can be acted on (*cf.* §3.3). In contrast, previous VLN agents can only explore currently perceived navigable viewpoints $\{o_{t,k}\}_t^{K_t}$ (Eq. 2), *i.e.*, the sub-nodes of current location. In addition, with SSM, the agent can easily change its current navigation direction by directly 'jumping' to another sub-node, which may be even observed several steps ago. Thus the agent gets rid of localized backtracking [28, 34] or explore-ahead [47] strategies.

Online Updating. SSM is online built and dynamically updated. At the beginning of an navigation episode, the node set is initialized as the start location. Later, SSM will be extended gradually. Assume that, at step t, an action, selected from $\mathcal{A}_t$, is taken to navigate a new place u (*i.e.*, a navigable sub-node). Then a new node will be added to represent u. Note that, in $\mathcal{G}_t$, there may exist one more sub-nodes correspond to u, as a same place can be observed from different viewpoints, and the agent does not know which sub-nodes correspond to u until u is navigated. After visiting u, the sub-nodes, in $\mathcal{G}_t$, that correspond to u can be found and removed; only the new added node will be left and connected to other navigable nodes. Then, SSM for the next navigation step is constructed, *i.e.*, $\mathcal{G}_{t+1}$.

3.2. Fine-Grained Instruction Grounding based Iterative Reasoning

Given current navigation state, a collect-read controller is learnt to read useful content from SSM for supporting next-step decision making. It automatically mines perception and object related textual information from instructions, and grounds them onto different elements of SSM. Then it makes use of iterative algorithms for long-range planning.

Perception-and-Action Aware Textual Grounding. The ability to understand natural-language instructions is critical to building intelligent VLN agents. The instructions interpret navigation routes by linguists, and thus can be grounded in perception and action, instead of arbitrary semantic tokens [9]. For example, instructions describe actions (*e.g.*, *turn left*, *walk forward*) closely relate to orientations, while instructions refer to landmarks in the environment (*e.g.*, *bedroom*, *table*) correspond more to visual perceptions. Whereas visual and orientation information are disentangled in our space representation, we separately explore perception and action related phases and ground them into different SSM elements. Inspired by [40], two different attention models are first learnt to assemble perception and action related textual features from $\boldsymbol{X}$, respectively:

$$\boldsymbol{x}_t^p = \text{att}^p(\boldsymbol{X}, \boldsymbol{h}_t) = \sum_l \alpha_l^p \boldsymbol{x}_l, \text{ where } \alpha_l^p = \text{softmax}_l(\boldsymbol{x}_l^\top \boldsymbol{W}_x^p \boldsymbol{h}_t);$$
$$\boldsymbol{x}_t^a = \text{att}^a(\boldsymbol{X}, \boldsymbol{h}_t) = \sum_l \alpha_l^a \boldsymbol{x}_l, \text{ where } \alpha_l^a = \text{softmax}_l(\boldsymbol{x}_l^\top \boldsymbol{W}_x^a \boldsymbol{h}_t). \quad (5)$$

Conditioned on current navigation state $\boldsymbol{h}_t$, the parsed perception and action related descriptions are of interest to the agent performing next-step navigation.

Next we generate perception and action aware states related to current navigation, respectively:

$$\boldsymbol{h}_t^p = \boldsymbol{W}_h^p[\boldsymbol{h}_t, \boldsymbol{x}_t^p]; \quad \boldsymbol{h}_t^a = \boldsymbol{W}_h^a[\boldsymbol{h}_t, \boldsymbol{x}_t^a]. \quad (6)$$

Then, for each node $v \in \mathcal{V}^t$, the collect-read controller assembles the visual information from its embedding $\boldsymbol{v}$, *i.e.*, visual features $\{\boldsymbol{f}_{v,k}\}_{k=1}^{K_v}$ of the panoramic observations, conditioned on the perception-aware state $\boldsymbol{h}_t^p$:

$$\hat{\boldsymbol{f}}_v = \text{att}^f(\boldsymbol{v}, \boldsymbol{h}_t^p) = \text{att}^f(\{\boldsymbol{f}_{v,k}\}_{k=1}^{K_v}, \boldsymbol{h}_t^p). \quad (7)$$

Similarly, for $v \in \mathcal{V}^t$, the collect-read controller assembles the orientation information from its outgoing edges, *i.e.*, $\{\boldsymbol{e}_{v,u}\}_{u \in \mathcal{N}_v}$, conditioned on the action-aware state $\boldsymbol{h}_t^a$:

$$\hat{\boldsymbol{r}}_v = \text{att}^r(\{\boldsymbol{e}_{v,u}\}_{u \in \mathcal{N}_v}, \boldsymbol{h}_t^a) = \text{att}^r(\{\boldsymbol{r}_{v,u}\}_{u \in \mathcal{N}_v}, \boldsymbol{h}_t^a), \quad (8)$$

where $\mathcal{N}_v$ denotes neighbors (*i.e.*, connected locations) of v.

Iterative Algorithm based Long-Term Reasoning. Then we follow classic iterative algorithms to perform long-term reasoning. It is achieved by a parametric *message passing* procedure, which iteratively updates the visual and orientation features by exchanging messages between nodes in the form of trainable functions. Specifically, at iteration s, $\hat{\boldsymbol{f}}_v$ and $\hat{\boldsymbol{r}}_v$ of each node v are respectively updated according to the received messages, *i.e.*, $\boldsymbol{m}_v^f$ and $\boldsymbol{m}_v^r$, which are information summarized from the neighbors $\mathcal{N}_v$:

$$\boldsymbol{m}_v^f = \sum_{u \in \mathcal{N}_v} \hat{\boldsymbol{f}}_u^s, \quad \hat{\boldsymbol{f}}_v^{s+1} = U^f(\boldsymbol{m}_v^f, \hat{\boldsymbol{f}}_v^s);$$
$$\boldsymbol{m}_v^r = \sum_{u \in \mathcal{N}_v} \hat{\boldsymbol{r}}_u^s, \quad \hat{\boldsymbol{r}}_v^{s+1} = U^r(\boldsymbol{m}_v^r, \hat{\boldsymbol{r}}_v^s), \quad (9)$$

where *update functions* $U^*(\cdot)$ are achieved by GRU. After S iterations of aggregation, improved visual and orientation features, *i.e.*, $\hat{\boldsymbol{f}}_v^S$ and $\hat{\boldsymbol{r}}_v^S$, are generated, through capturing the context within the S-hop neighborhood of node v.

3.3. Frontier-Exploration based Decision Making

Global Action Space based One-Step Decision Making.
SSM provides a global action space $\mathcal{A}_t = \{o_{v,k}\}_{v=1,k=1}^{|\mathcal{V}_t|,K_v}$, comprised of all the navigable sub-nodes in $\mathcal{G}_t$. Each sub-node $o_{v,k}$ is associated with the visual $\boldsymbol{f}_{v,k}$ and orientation $\boldsymbol{r}_{v,k}$ features of its corresponding viewpoint. We first formulate its confidence of being the next navigation action according to visual and orientation cues, respectively:

$$q_{v,k}^f = [\boldsymbol{f}_{v,k}, \hat{\boldsymbol{f}}_v^S]^\top \boldsymbol{W}_f^q \boldsymbol{h}_t^p, \quad q_{v,k}^r = [\boldsymbol{r}_{v,k}, \hat{\boldsymbol{r}}_v^S]^\top \boldsymbol{W}_q^r \boldsymbol{h}_t^a. \quad (10)$$

The perception-aware confidence $q_{v,k}^f$ (action-aware confidence $q_{v,k}^r$) is computed by considering the visual (orientation) cues of itself and its parent node. Then the final confidence is computed as a weighted sum of $q_{v,k}^f$ and $q_{v,k}^r$ [40]:

$$p(o_{v,k}) = \text{softmax}_{v,k}([q_{v,k}^f, q_{v,k}^r]^\top \boldsymbol{w}_q^p), \quad (11)$$

where $\boldsymbol{w}_q^p$ is the combination weight: $\boldsymbol{w}_q^p = \boldsymbol{W}^w[\boldsymbol{h}_t^p, \boldsymbol{h}_t^a]$.

Finally, a feasible navigation action a_t is selected from $\mathcal{A}_t$, according to:

$$a_t = \underset{v,k}{\text{argmax}} \ p(o_{v,k}). \quad (12)$$

However, as $\mathcal{A}_t$ will become large after several navigation steps, this makes policy learning hard and converge slow in practice. To address this issue, we propose a frontier-exploration based decision making strategy.

Navigation Utility Driven Frontier Selection. We consider frontiers to be points at the boundary of known regions of a map [53]. In $\mathcal{G}_t$, the frontiers $\mathcal{W}_t \subset \mathcal{V}_t$ are the nodes with more than one navigable sub-nodes, *i.e.*, $\mathcal{W}_t = \{w | w \in \mathcal{V}_t, K_w \geq 1\}$. In Fig. 2(c), the frontier nodes are highlighted in green. Frontiers can be viewed important points to facilitate exploration in a partially observed environment, as no matter which sub-node is selected to be navigated, the navigation must starts at a certain frontier node. Therefore, we define such frontiers as sub-goals for navigation decision making. Instead of directly selecting an action from $\mathcal{A}_t$, we first find a frontier node (Fig. 2(d)) and then choose one of its sub-nodes (Fig. 2(e)) to navigate.

At step t, a frontier node $w \in \mathcal{W}_t$ is selected according to $\text{argmax}_w \ p(w)$, where $p(w)$ is computed as:

$$c_w^f = \hat{\boldsymbol{f}}_w^{S\top} \boldsymbol{W}_f^c \boldsymbol{h}_t^p, \quad c_w^r = \hat{\boldsymbol{r}}_w^{S\top} \boldsymbol{W}_r^c \boldsymbol{h}_t^a,$$
$$p(w) = \text{softmax}_w([c_w^f, c_w^r]^\top \boldsymbol{w}_c^p). \quad (13)$$

The agent can easily make error-correction: performing even long-term backtrack by only one-step frontier selection. Our frontier selection policy is learnt in an *navigation utility driven* manner. During RL based online policy learning, the agent is rewarded at each frontier selection step for getting closer to the navigation destination (*cf.* §3.4). This is different to previous robot control methods selecting frontiers based on certain heuristics [53, 36, 1], such as maximizing information gain (the amount of new information that can be acquired) or minimizing moving cost.

Frontier Sub-Node Navigation. If the intended frontier w is not the current position of the agent, the agent will move to w along the shortest path over $\mathcal{G}_t$. The navigation states, *i.e.*, $\boldsymbol{h}_t, \boldsymbol{h}_t^p$, and $\boldsymbol{h}_t^a$, will be also updated by the observations on-the-move. Otherwise, the agent will stay at its current position and maintain current navigation states unchanged. Then, the moving action a_t is selected from the navigable sub-nodes $\{o_{w,k}\}_{k=1}^{K_w}$ of the selected frontier w, according to $\text{argmax}_k \ p(o_{w,k})$. Here $p(o_{w,k})$ is computed by:

$$q_{w,k}^f = \boldsymbol{f}_{w,k} \boldsymbol{W}_f^q \boldsymbol{h}_t^p, \quad q_{w,k}^r = \boldsymbol{r}_{w,k} \boldsymbol{W}_r^q \boldsymbol{h}_t^a,$$
$$p(o_{w,k}) = \text{softmax}_k([q_{w,k}^f, q_{w,k}^r]^\top \boldsymbol{w}_q^p). \quad (14)$$

By performing a_t, the agent will reach a new position u (*i.e.*, $\text{argmax}_{w,k} \ p(o_{w,k})$). Then, u will be added into SSM, forming $\mathcal{G}_{t+1}$ (*cf.* §3.1). The agent will also update its navigation state $\boldsymbol{h}_t$ with its current panoramic observation $\boldsymbol{O}_{t+1}$ (*i.e.*, $\boldsymbol{O}_u$) through Eq. 1. With the new SSM (*i.e.*, $\mathcal{G}_{t+1}$) and navigation state (*i.e.*, $\boldsymbol{h}_{t+1}$), the agent will make next-round navigation. Above process will be executed until the agent thinks it has reached the destination. See §4.3 for the empirical study of the effectiveness of our frontier-exploration based decision making strategy.

3.4. Implementation Details

Network Architecture: We build upon a basic navigation architecture in [45]. As in [3, 48, 50], each panoramic view is divided into 36 sub-views, *i.e.*, $12 \text{ headings} \times 3 \text{ elevations}$ with 30 degree intervals. For each viewpoint $o_{t,k}$, the visual embedding $\boldsymbol{f}_{t,k}$ is a 2048-d ResNet-152 [20] feature and orientation embedding $\boldsymbol{r}_{t,k}$ is a 128-d vector (*i.e.*, 4-d orientation feature $(\cos \theta_{t,k}, \sin \theta_{t,k}, \cos \phi_{t,k}, \sin \phi_{t,k})$ are tiled 32 times). Instruction embeddings $\boldsymbol{X}$ are obtained from an LSTM with a 512 hidden size.

Network Training: Following [56, 50, 45, 48], our model is trained using both imitation learning (IL) and reinforcement learning (RL). For IL, we adopt both teacher forcing and student forcing to train our agent. In teacher forcing, the agent learns the offline policy through behavior cloning of the expert navigation trajectories. However, the agent is biased to the teacher action and limits exploration to a local action space that is the in ground-truth shortest-path trajectory. In student forcing, the agent samples its next-step action from its predicted probability distribution. We choose the node on SSM that is closest to the target location as the teacher action. In RL based online policy learning, the rewards for frontier and action selection are designed in a similar spirit. At each frontier or navigation action selection step, the agent receives an immediate reward, *i.e.*, the change in the distance to the goal destination. At the end of the episode, the agent receives a reward only if it terminated successfully [48, 45]. In addition, following [13, 45, 40, 11, 28], back translation [13] based data augmentation technique is used for R2R dataset [3].

Models	R2R Dataset														
	validation seen					validation unseen					test unseen				
	SR↑	NE↓	TL↓	OR↑	SPL↑	SR↑	NE↓	TL↓	OR↑	SPL↑	SR↑	NE↓	TL↓	OR↑	SPL↑
Student-Forcing [3]	0.39	6.01	11.3	0.53	-	0.22	7.81	8.39	0.28	-	0.20	7.85	**8.13**	0.27	0.18
RPA [50]	0.43	5.56	**8.46**	0.53	-	0.25	7.65	7.22	0.32	-	0.25	7.53	9.15	0.33	0.23
E-Dropout [45]	0.55	4.71	10.1	-	0.53	0.47	5.49	9.37	-	**0.43**	-	-	-	-	-
Regretful [34]	0.65	3.69	-	0.72	**0.59**	0.48	5.36	-	0.61	0.37	-	-	-	-	-
EGP [11]	-	-	-	-	-	0.52	5.34	-	0.65	0.41					
Active Perception [47]	**0.66**	**3.35**	19.8	**0.79**	0.51	0.55	**4.40**	19.9	**0.70**	0.40	0.56	4.77	21.0	**0.73**	0.37
Ours	0.65	3.77	13.5	0.74	0.57	**0.56**	4.88	18.9	0.69	0.42	**0.57**	**4.66**	18.5	0.68	**0.44**
Speaker-Follower [13]*	0.66	3.36	-	0.74	-	0.36	6.62	-	0.45	-	0.35	6.62	14.8	0.44	0.28
RCM [48]*	0.67	3.53	**10.7**	0.75	-	0.43	6.09	11.5	0.50	-	0.43	6.12	12.0	0.50	0.38
Self-Monitoring [33]*	0.67	3.22	-	0.78	0.58	0.45	5.52	-	0.56	0.32	0.43	5.99	18.0	0.55	0.32
Regretful [34]*	0.69	3.23	-	0.77	0.63	0.50	5.32	-	0.59	0.41	0.48	5.69	13.7	0.56	0.40
E-Dropout [45]*	0.62	3.99	11.0	-	0.59	0.52	5.22	**10.7**	-	0.48	0.51	5.23	**11.7**	0.59	0.47
OAAM* [40]	0.65	-	10.2	0.73	0.62	0.54	-	9.95	0.61	0.50	0.53	-	10.4	0.61	0.50
EGP* [11]	-	-	-	-	-	0.56	4.83	-	0.64	0.44	0.53	5.34	-	0.61	0.42
Tactical Rewind [28]*	-	-	-	-	-	0.56	4.97	21.2	-	0.43	0.54	5.14	22.1	0.64	0.41
AuxRN [55]*	0.70	3.33	-	0.78	**0.67**	0.55	5.28	-	0.62	**0.50**	0.55	5.15	-	0.62	**0.51**
Active Perception [47]*	0.70	3.20	19.7	**0.80**	0.52	0.58	4.36	20.6	0.70	0.40	0.60	**4.33**	21.6	**0.71**	0.41
Ours*	**0.71**	**3.10**	14.7	**0.80**	0.62	**0.62**	**4.32**	20.7	**0.73**	0.45	**0.61**	4.57	20.4	0.70	0.46

Table 1: Quantitative comparison results (§4.1) on R2R dataset [3]. For compliance with the evaluation server, we report SR as fractions. '*': back translation based data augmentation. '−': unavailable statistics.

(a) Groundtruth VLN path in a top-down view

(b) Basic agent

(c) Our agent

(d) Panoramic observation at P1
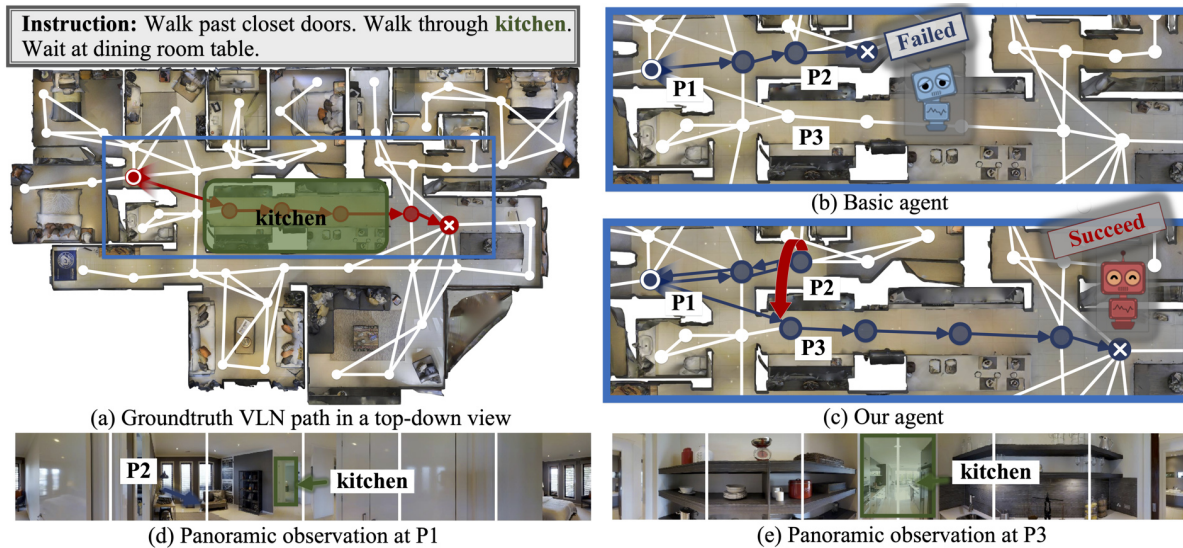
(e) Panoramic observation at P3

Figure 3: A representative visual result on R2R dataset [3] (§4.1). Due to partial observability of agent's perception system, it is hard to find kitchen at position P1 (see (d)). Thus the basic agent, in (b), goes the wrong way and ends in failure. However, in (c), our agent easily turns back and explores another direction (*i.e.*, P3). At P3, the kitchen is easy to find (see (e)) and the instruction is successfully completed.

Reproducibility: Our model is implemented in PyTorch and trained on 8 NVIDIA Tesla V100 GPUs. To reveal full details of our method, our implementations are released.

4. Experiment

4.1. Performance on R2R Dataset

Dataset. We first conduct experiments on R2R dataset [3] which is split into four sets: training (61 environments, 14,039 instructions), validation seen (61 environments, 1,021 instructions), validation unseen (11 environments, 2,349 instructions), and test unseen (18 environments, 4,173 instructions). There are no overlapping environments between the unseen and training sets.

Evaluation Metric. Following [3, 13], five metrics are used: (1) *Success Rate* (SR), the primary metric, considers the percentage of final positions less than 3 m away from the goal location. (2) *Navigation Error* (NE) refers to the shortest distance between agent's final position and the goal location. (3) *Trajectory Length* (TL) measures the total length of agent trajectories. (4) *Oracle success Rate* (OR) is the success rate if the agent can stop at the closest point to the goal along its trajectory. (5) *Success rate weighted by Path Length* (SPL) [2] is a trade-off between SR and TL.

Quantitative Result. Table 1 compares our model against 12 state-of-the-art VLN models on R2R dataset. We find that our model outperforms other competitors across most metrics. For instance, compared with current top-leading method, Active Perception [47], our model performs better

Models	R4R Dataset											
	validation seen						validation unseen					
	NE↓	TL↓	SR↑	CLS↑	nDTW↑	SDTW↑	NE↓	TL↓	SR↑	CLS↑	nDTW↑	SDTW↑
Speaker-Follower [13]	5.35	15.4	0.52	0.46	-	-	8.47	19.9	0.24	0.30	-	-
RCM+goal oriented [48]	5.11	24.5	0.56	0.40	-	-	8.45	32.5	0.29	0.20	0.27	0.11
RCM+fidelity oriented [48]	5.37	18.8	0.53	0.55	-	-	8.08	28.5	0.26	0.35	0.30	0.13
PTA low-level [30]	5.11	11.9	0.57	0.52	0.42	0.29	8.19	**10.2**	0.27	0.35	0.20	0.08
PTA high-level [30]	**4.54**	16.5	0.58	0.60	**0.58**	0.41	8.25	17.7	0.24	0.37	0.32	0.10
EGP [11]	-	-	-	-	-	-	**8.00**	18.3	0.30	0.44	0.37	0.18
E-Drop [45]	-	19.9	0.52	0.53	-	0.27	-	27.0	0.29	0.34	-	0.09
OAAM [40]	-	**11.8**	0.56	0.54	-	0.32	-	13.8	0.31	0.40	-	0.11
Ours	4.60	19.4	**0.63**	**0.65**	0.56	**0.44**	8.27	22.1	**0.32**	**0.53**	**0.39**	**0.19**

Table 2: Quantitative comparison results (§4.2) on R4R dataset [26]. '−': unavailable statistics.

Instruction: 1. Go straight until you exit the room. Continue going straight and wait at the entrance to kitchen. 2. Walk towards oven and turn right through doorway. Go down hallway and turn left and wait at bathroom.

(a) Groundtruth VLN path in a top-down view   (b) Basic agent   (c) Our agent
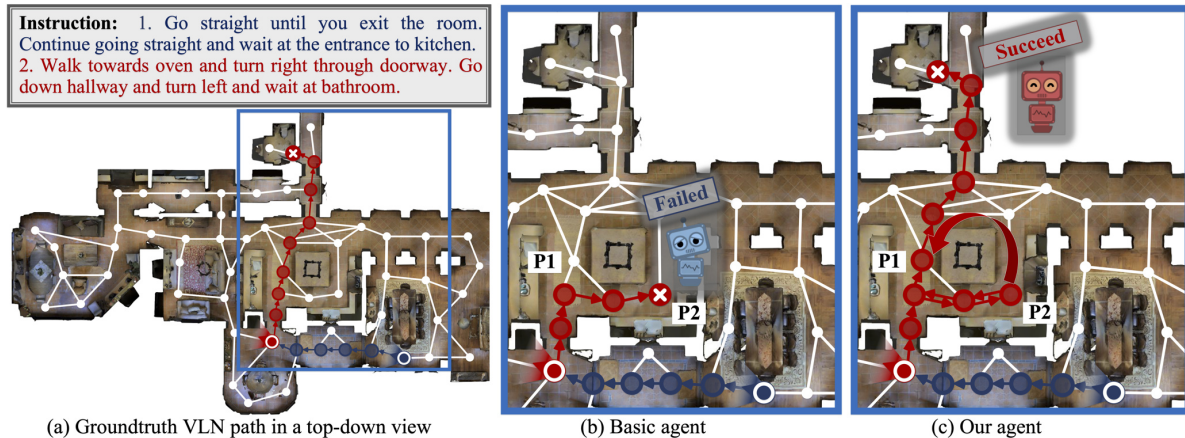
Figure 4: A representative visual result on R4R dataset [26] (§4.2). The instruction "*Go straight* · · ·" is easy to follow and corresponding navigation trajectories are highlighted in blue. However, in (b), the basic agent is confused by the ambiguous instruction "· · · *turn right through doorway* · · ·", causing failed navigation. Through SSM, our agent can easily access to past observations and make more robust navigation. As shown in (c), our agent successfully returns to the correct direction and reaches the target location finally.

in term of SR, and achieves a huge boost in SPL, across `validation seen`, `validation unseen`, and `test unseen`. Our model also significantly surpasses the second best method, AuxRN [55], in terms of SR, NE, TL and OR, without using any auxiliary supervision signals. Despite having similar motivation of exploring global, structured action space, EGP [11] achieves an SR of 0.53 on `test unseen` while our model yields 0.61. We also notice that our agent without data augmentation already outperforms existing methods on SR, NE and OR, only excepting [47].

Visual Result. As shown in Fig. 3(a), *kitchen* is a critical landmark for instruction execution. However, at the start location (*i.e.*, P1), *kitchen* is hard to be observed (Fig. 3(d)). Thus the basic agent navigates a wrong direction and finally fails (Fig. 3(b)). However, with the help of SSM, our agent can make long-range planning over all the explored environment and quickly shift its navigation direction from P2 to P3 (Fig. 3(c)). As shown in Fig. 3(e), our agent then finds *kitchen* and accomplishes the instruction successfully.

4.2. Performance on R4R Dataset

Dataset. R4R [26] is an extended version of R2R with longer instructions and trajectories. R4R has 278, 766 in-

structions and contains three splits: `training` (61 environments, 233, 613 instructions), `validation seen` (61 environments, 1, 035 instructions), and `validation unseen` (11 environments, 45, 162 instructions).

Evaluation Metric. Following [48, 30, 11], six evaluation metrics are used for R4R. Specifically, in addition to (1) SR, (2) NE, and (3) TL, three new metrics are introduced: (4) *Coverage weighted by Length Score* (CLS) [26] that considers both how well the ground truth path is covered by the agent trajectory and the trajectory length, (5) *normalized Dynamic Time Warping* (nDTW) that measures the fidelity (order consistency) of agent trajectories, and (6) *Success rate weighted normalized Dynamic Time Warping* (SDTW) that stresses the importance of reaching the goal comparing to nDTW. Here CLS is considered as the primary metric.

Quantitative Result. Table 2 presents comparisons with six top-leading VLN models on R4R dataset. We can find that our approach sets new state-of-the-arts for most metrics. Notably, on `validation seen` and `validation unseen`, our model yields CLSs of 0.65 and 0.53, respectively, while those for the second best method are 0.60 and 0.44. Our model also shows significant performance gains in terms of SR, nDTW, and SDTW. In addition, among all

| Model | SSM Component | R2R Dataset | | | | | | | | | |
| | | validation seen | | | | | validation unseen | | | | |
		SR↑	NE↓	TL↓	OR↑	SPL↑	SR↑	NE↓	TL↓	OR↑	SPL↑
Basic agent [45]	-	0.62	3.99	11.0	0.71	0.59	0.52	5.22	10.7	0.58	0.48
Full model	**Fine-grained instruction grounding + Iterative reasoning + Frontier-exploration based decision making**	0.71	3.10	14.7	0.80	0.62	0.62	4.32	20.7	0.73	0.45
Variants	Full model *w/o.* fine-grained instruction grounding	0.70	3.44	13.5	0.80	0.62	0.59	4.57	20.3	0.72	0.43
	Full model *w/o.* iterative reasoning	0.69	3.51	12.9	0.78	0.61	0.57	4.60	19.8	0.70	0.42
	Full model *w.* local action space based decision making	0.67	3.62	11.9	0.75	0.63	0.54	5.21	12.6	0.64	0.49
	Full model *w.* global action space based one-step decision making	0.68	3.25	13.2	0.74	0.60	0.56	4.76	19.8	0.68	0.41

Table 3: Ablation study (§4.3) on `validation seen` and `validation unseen` sets of R2R dataset [3].

the methods, our agent achieves smallest performance gap between seen and unseen environments, *e.g.*, 0.12 in terms of CLS. This evidences the advantage of SSM that helps our agent learn more generalized navigation policy. Note that [22] is excluded from the comparisons on R2R and R4R, as it makes use of extra data and object-level cues.

Visual Result. Fig. 4(a) gives a challenging case in R4R. There are two conjuncted instructions, corresponding to twisted routes connecting two shortest-path trajectories in R2R, are highlighted in blue and red, respectively. After successfully accomplishing the first instruction "*Go straight··*", the basic agent is misled by the ambiguous instruction "··· *turn right through doorway··*" and mistakenly stops at position P2 (Fig. 4(b)). However, SSM allows our agent to make more robust decision on the global action space. As shown in Fig. 4(c), our agent easily returns back to the correction direction (*i.e.*, P1) and finally succeeds.

4.3. Diagnostic Experiments

For thoroughly assessing the efficacy of essential components of our approach, a set of ablation studies are conducted on the two validations sets of R2R dataset [3].

Whole SSM Architecture Design. As shown in the first two rows in Table 3, our "Full model" significantly outperforms the "Basic Agent" [45] on both `validation seen` and `validation unseen` sets, across all metrics. This demonstrates the effectiveness of our whole model design.

Perception-and-Action Aware Instruction Grounding. In SSM, visual and geometric information can be disentangled (§3.1). This powerful scene representation allows our agent to better interpret instructions and predict more accurate navigation actions (§3.2). To verify this point, we only use the original, perception-and-action agnostic navigation state h_t, instead of perception and action aware states (*i.e.*, h_t^p and h_t^a in Eq. 6), for visual and geometric information assembling (Eqs.7-8) and action probability estimation (Eqs. 13-14). Considering the second and third rows in Table 3, we can conclude that visual and geometric information disentangled scene representation with fine-grained instruction grounding indeed boosts VLN performance.

Iterative Reasoning. In §3.2, a parametric message passing procedure is conducted over our SSM, which step-by-step gathers information for long-range planning and decision-making supporting. After omitting this procedure, we obtain a baseline "Full model *w/o.* iterative reasoning". Comparing this baseline with our "Full model", significant performance drop can be observed (see Table 3).

Frontier-Exploration based Decision Making. Our SSM provides a global navigation action space – the agent is able to navigate any past visited location when necessary (§3.3). To demonstrate this advantage, we develop a new baseline "Full model *w.* local action space based decision making", *i.e.*, the agent is only allowed to navigate current visible directions. This strategy is widely adopted in current VLN models. As evidenced in Table 3, the performance of the new baseline is significantly worse than "Full model', revealing the necessity of considering a global action space. To erase the difficulty of policy learning on the fast-growing global action space, we develop a frontier-exploration based decision making strategy. Compared with directly making decision on the whole global action space (*i.e.*, "Full model *w.* global action space based one-step decision making"), our strategy obtains a consistent margin across all metrics.

5. Conclusion

Memory and mapping are crucial components to enabling intelligent navigation in partially observable environments. However, current VLN agents are simply built upon recurrent neural networks. Their latent network state based implicit memory is unsuitable for modeling long-term structural data dependencies, hence limiting their planning ability. In this paper, we develop a structured, explicit memory architecture, SSM, that allows VLN agents to access to its past percepts and explore environment layouts. With this expressive and persistent space representation, our agent shows advantages in fine-grained instruction grounding, long-term reasoning, and global decision-making. We demonstrate empirically that our SSM based VLN agent sets state-of-the-arts on challenging R2R and R4R datasets.

References

[1] Francesco Amigoni. Experimental evaluation of some exploration strategies for mobile robots. In *ICRA*, 2008. 5

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

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1, 2, 3, 5, 6, 8

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 2

[5] John Canny. *The Complexity of Robot Motion Planning*. MIT press, 1988. 2

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *3DV*, 2017. 1

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 1, 2

[8] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020. 1, 2

[9] David L Chen and Raymond J Mooney. Learning to interpret natural language navigation instructions from observations. In *AAAI*, 2011. 1, 4

[10] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, Fei Xia, Alvaro Soto, Marynel Vázquez, and Silvio Savarese. A behavioral approach to visual navigation with graph localization networks. *arXiv preprint arXiv:1903.00445*, 2019. 2

[11] Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. In *NeurIPS*, 2020. 2, 5, 6, 7

[12] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, 2019. 1

[13] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018. 1, 2, 3, 5, 6, 7

[14] Tsu-Jui Fu, Xin Wang, Matthew Peterson, Scott Grafton, Miguel Eckstein, and William Yang Wang. Counterfactual vision-and-language navigation via adversarial path sampling. In *ECCV*, 2020. 1, 2

[15] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015. 1

[16] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 2

[17] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017. 1, 2

[18] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. *CVPR*, 2020. 2

[19] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[22] Yicong Hong, Cristian Rodriguez-Opazo, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. *NeurIPS*, 2020. 2, 8

[23] Yicong Hong, Cristian Rodriguez-Opazo, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. In *ACL*, 2020. 2

[24] Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. Are you looking? grounding to multiple modalities in vision-and-language navigation. In *ACL*, 2019. 1

[25] Haoshuo Huang, Vihan Jain, Harsh Mehta, Alexander Ku, Gabriel Magalhaes, Jason Baldridge, and Eugene Ie. Transferable representation learning in vision-and-language navigation. In *ICCV*, 2019. 1, 2

[26] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In *ACL*, 2019. 2, 7

[27] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. 2

[28] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *CVPR*, 2019. 1, 2, 4, 5, 6

[29] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *ECCV*, 2020. 2

[30] Federico Landi, Lorenzo Baraldi, Marcella Cornia, Massimiliano Corsini, and Rita Cucchiara. Perceive, transform, and act: Multi-modal attention networks for vision-and-language navigation. *arXiv preprint arXiv:1911.12377*, 2020. 7

[31] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions*, (5):293–308, 2001. 2

[32] Lisa Lee, Emilio Parisotto, Devendra Singh Chaplot, Eric Xing, and Ruslan Salakhutdinov. Gated path planning networks. In *ICML*, 2018. 2

[33] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019. 1, 2, 6

[34] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided

navigation through progress estimation. In *CVPR*, 2019. 1, 2, 4, 6

[35] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *ECCV*, 2020. 2

[36] Alexei A Makarenko, Stefan B Williams, Frederic Bourgault, and Hugh F Durrant-Whyte. An experiment in integrated exploration. In *IROS*, 2002. 5

[37] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, 2016. 1

[38] Junhyuk Oh, Valliappa Chockalingam, Honglak Lee, et al. Control of memory, active perception, and action in minecraft. In *ICML*, 2016. 2

[39] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *ICLR*, 2019. 2

[40] Yuankai Qi, Zizheng Pan, Shengping Zhang, Anton van den Hengel, and Qi Wu. Object-and-action aware model for visual language navigation. In *ECCV*, 2020. 1, 2, 4, 5, 6, 7

[41] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 2

[42] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 2

[43] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995. 2

[44] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NeurIPS*, 2015. 2

[45] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL*, 2019. 1, 2, 3, 5, 6, 7, 8

[46] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 1, 2

[47] Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. Active visual information gathering for vision-language navigation. In *ECCV*, 2020. 1, 2, 3, 4, 6, 7

[48] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019. 1, 2, 5, 6, 7

[49] Xin Wang, Vihan Jain, Eugene Ie, William Yang Wang, Zornitsa Kozareva, and Sujith Ravi. Environment-agnostic multitask learning for natural language grounded navigation. In *ECCV*, 2020. 1, 2

[50] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *ECCV*, 2018. 1, 2, 5, 6

[51] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015. 2

[52] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. Bayesian relational memory for semantic visual navigation. In *ICCV*, 2019. 2

[53] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, 1997. 5

[54] Jingwei Zhang, Lei Tai, Joschka Boedecker, Wolfram Burgard, and Ming Liu. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017. 2

[55] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020. 1, 2, 6, 7

[56] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 5