mathematics

MDPI

Article

Self-Organizing Memory Based on Adaptive Resonance Theory for Vision and Language Navigation

Wansen Wu, Yue Hu *, Kai Xu, Long Qin and Quanjun Yin

College of Systems Engineering, National University of Defense Technology, Changsha 410073, China; wuwansen14@nudt.edu.cn (W.W.); xukai09@nudt.edu.cn (K.X.); qldbx2007@sina.com (L.Q.); yin_quanjun@163.com (Q.Y.)

* Correspondence: huyue11@nudt.edu.cn

Abstract: Vision and Language Navigation (VLN) is a task in which an agent needs to understand natural language instructions to reach the target location in a real-scene environment. To improve the model ability of long-horizon planning, emerging research focuses on extending the models with different types of memory structures, mainly including topological maps or a hidden state vector. However, the fixed-length hidden state vector is often insufficient to capture long-term temporal context. In comparison, topological maps have been shown to be beneficial for many robotic navigation tasks. Therefore, we focus on building a feasible and effective topological map representation and using it to improve the navigation performance and the generalization across seen and unseen environments. This paper presents a S elf-organizing Memory based on Adaptive Resonance Theory (SMART) module for incremental topological mapping and a framework for utilizing the SMART module to guide navigation. Based on fusion adaptive resonance theory networks, the SMART module can extract salient scenes from historical observations and build a topological map of the environmental layout. It provides a compact spatial representation and supports the discovery of novel shortcuts through inferences while being explainable in terms of cognitive science. Furthermore, given a language instruction and on top of the topological map, we propose a vision–language alignment framework for navigational decision-making. Notably, the framework utilizes three off-the-shelf pre-trained models to perform landmark extraction, node–landmark matching, and low-level controlling, without any fine-tuning on human-annotated datasets. We validate our approach using the Habitat simulator on VLN-CE tasks, which provides a photo-realistic environment for the embodied agent in continuous action space. The experimental results demonstrate that our approach achieves comparable performance to the supervised baseline.

Keywords: vision and language navigation; adaptive resonance theory; mapping

MSC: 68T05; 68T30; 68T45

Citation: Wu, W.; Hu, Y.; Xu, K.; Qin, L.; Yin, Q. Self-Organizing Memory Based on Adaptive Resonance Theory for Vision and Language Navigation. *Mathematics* **2023**, *11*, 4192. https://doi.org/10.3390/math11194192

Academic Editors: Guangwei Gao, Juncheng Li and Zhi Li

Received: 16 September 2023
Revised: 1 October 2023
Accepted: 6 October 2023
Published: 7 October 2023

1. Introduction

Vision and language navigation plays a crucial role in embodied AI research, which requires an agent to interpret natural language navigational commands in light of photo-realistic images generated by a previously unseen scene [1]. For example, the agent is given an instruction at a start position like *"Exit the room then turn hard left, walk forward down the stairs and stop at the front of the bathroom."* The agent can obtain the surrounding environment as a visual observation at each step and is expected to make a decision for the next step until it reaches the target.

Since the agent is supposed to perform tasks over long horizons in previously unseen environments, it is necessary to take advantage of past experience. Numerous studies proposed using external memory structure to improve generalization in unseen environments, such as topological maps [2–4] and a hidden state vector based on recurrent neural network [5–7], which show remarkable improvements in the VLN problem.

However, the hidden state vector memory works well in discrete environments (i.e., where the agent can teleport between pre-defined discrete navigable points), but struggles in continuous environments (i.e., where the agent needs to perform low-level actions, such as moving forward 0.25 meters or turning left 15 degrees) [8]. In continuous environments, the action and state space of the agent become very large and the action sequence is usually 10 times longer than that in a discrete environment (55.88 compared to 5 on average). A fixed-length recurrent vector can hardly capture the long-term dependency among states and actions. As an alternative, many recent works have proposed to build an expanding topological graph as the spatial memory while the agent traverses the environment [2,4]. Such topological-graph-based models primarily rely on training a neural network to discern the similarity of dense trajectory nodes to sparsify and build the map. However, this approach becomes increasingly complex with continuous exploration and is unable to establish shortcuts between nodes, rendering it highly inefficient.

Extensive psychology and cognitive science research has shown that humans perform navigation tasks by recalling recognizable places throughout their journey from their memory instead of from a metric knowledge of space [9,10]. However, some graph-based works like [3] often have linear complexity that grows with the exploration duration and learn too many nodes in the topological map due to the indiscriminate treatment of all observations. Such approaches run contrary to much more compact, scalable, and flexible spatial memory, wherein salient landmarks rather than trivial observations are encoded [11]. Moreover, a map that is too dense may provide less navigational guidance and cause the agent to have many scrubbed and therefore irrational behaviors during the navigation process.

On the contrary, the hippocampus in human brains can encode space-related trajectories from contextual information related to events in episodic memory, which identifies salient scenes and navigation-related vantage points [12]. Inspired by this observation, we present Self-organizing Memory based on Adaptive Resonance Theory (SMART) to address the aforementioned problem. Particularly, SMART constructs spatial memory as a topological map based on a unified set of fuzzy operations in a class of self-organizing neural networks known as fusion adaptive resonance theory (ART) networks [13,14]. An ever-extending trajectory is input into the memory module, wherein an observation is resonated with an existing memory template or is regarded as a new event representing the surrounding spatial–temporal context. A sequence of such separate events is extracted as an episode from streaming observations. A graph of event nodes is gradually constructed to specify the spatial layout of the traversed environment.

Apart from encoding spatial memory, the other advantage that topological representations endow a VLN agent with is the cross-modal alignment between vision and language. Specifically, the given instructions are naturally anchored on discrete locations or objects, such as “the room” and “the stairs”, instead of being based on continuously changing observations. With a sparsified topology, this paper proposes a hierarchical navigation framework to match the landmark descriptions in the instruction and the visual nodes in the graph so that the agent can more precisely know its next subgoal. To implement this idea, we leverage three off-the-shelf pre-trained models from previous works to perform landmark extraction, node–landmark matching and continuous action control. Along with the memory module, the proposed approach requires no human-labeled data nor no additional training, meaning it is a zero-shot paradigm. We evaluate the proposed method on popular a continuous VLN environment named VLN-CE built on the Habitat simulator [15]. We conduct an empirical comparison between SMART, CMTP [4], a recently proposed topological graph based method in VLN-CE, and several discrete–continuous transferred models. The results show that SMART has much better performance than CMTP and stronger generalization across unseen environments than other baselines. The main contributions of this paper are as follows:

- We present the Self-organizing Memory based on the Adaptive Resonance Theory (SMART) module to address the topological mapping problem in a continuous state and action space.
- We also present a navigation framework that can utilize SMART to perform vision–language alignment for improving navigation performance, which incorporates three independent off-the-shelf pre-trained models without any fine-tuning.
- We conducted extensive experiments to validate the effectiveness of the SMART model. Experimental results show our approach has promising performance on the val-seen and test set, even compared against supervised models based on large-scale training, and show a good generalization performance.

2. Related Work

2.1. Vision and Language Navigation

VLN tasks require an embodied agent to follow natural language instructions to navigate in previously unseen photo-realistic environments using visual observations [16]. Most of the systems for solving VLN tasks use an end-to-end based method [1]. In addition, other works focus on utilizing pre-trained models [6,7], global exploration strategies [2,17], auxiliary losses [18,19], and effective language grounding methods [20] to improve the model's performance. However, the task setting of VLN allows the agent to teleport between navigable points, which is far away from the robot navigation in real-world environments. In [8], they propose VLN-CE, where the agent can move freely in the continuous environment through several low-level actions. This is a more challenging task with notably longer action steps for each navigation episode compared with VLN. Moreover, the free-moving environment requires the agent to have the ability to bypass obstacles and avoid collisions, which are issues that have not been considered in discrete environments. To address this problem, we propose a two-stage navigation framework, where the agent builds the topological graph of the environment in the exploration stage and then performs navigation with the topological graph in the navigation stage. The agent can use a topological graph for long-horizon reasoning while using a navigation model for short-horizon controlling.

Another challenge in the VLN task is that a large amount of labeled data is desired for training the supervised model. This kind of model learns navigational policy directly from the vision and language, which performs well in the seen scenes but degrades significantly in the previously unseen scenes. As a result of recent developments in large-scale natural language and image models trained on diverse datasets, applications in a wide range of textual [21], visual [22], and embodied areas [7] have become feasible. The pre-trained models have shown great few-shot generalization capabilities and make it possible to solve many downstream tasks without fine-tuning. Inspired by this, we try to use the pre-trained model to directly solve the VLN task, in order to avoid the problem of insufficient training data and to improve generalization performance.

2.2. Spatial Memory

Spatial memory is an important concept in biological research and has received extensive attention since Tolman et al. [23] proposed the concept of the cognitive map. Specifically, humans extract perceptually salient landmarks from traversed routes and build a sketch of the environment with those landmarks and the reachability between pairs of them. Moreover, the hippocampus is also an important part of building episodic memory [24], and the spatiotemporal contexts of episodic memory can be further transferred and consolidated. Inspired by this, many works on spatial cognition discuss the importance of landmarks on spatial representation using the cognitive map in navigation tasks [25–28]. In the field of VLN, Hong et al. [29] demonstrate that modeling the relationships between the scene and salient landmarks can improve the performance of the navigation agent. In our work, we implicitly integrate the perceptual and cognitive salience into the learning process by the clustering and generalization ability of ART.

2.3. Existing Memory Architectures in VLN

Memory is an important capability that can improve the performance of different domains in artificial intelligence, such as natural language processing, computer vision, and robot navigation [30,31]. Specifically, in the field of robot navigation, memory architectures are usually divided into two categories, i.e., unstructured and structured memory. Unstructured memory representations are the fixed-length latent states of RNN or LSTM and have been widely used in many works in VLN [1,19]. However, unstructured memory suffers from the problem of long-term dependencies in large or complex environments. In comparison, structured memory can represent the environmental layout in an explicit way, such as metric and topological graphs. For example, [32] use the metric semantic spatial map to record the geometry of the environment, and [33] build a multi-scale metric map by an end-to-end system in a visual navigation task. However, in a continuous environment, building and maintaining a precise metric map from the original sensory signals is inefficient. In addition, as discussed above, spatial models that generate a global metric map are unreliable because of their lack of biological interpretation and weak robustness to noisy data. In this paper, we use a topological graph as memory to aid navigation because it has been proven to be effective in many navigation tasks [26,34]. Moreover, topological graphs are more convenient to build and update due to their sparse representation in a continuous space.

3. Method

VLN-CE is a navigation task in a continuous 3D environment where agents follow natural language navigation directions by performing low-level actions. Specifically, the environment is continuous and the agent must execute a series of low-level actions to find the goal instead of teleporting among the discrete navigable points; therefore, VLN-CE is a more realistic setting [8]. At the beginning of each episode, the agent receives a natural language instruction denoted as L. During navigation, at step t, the agent observes the RGB-D image of the environment and makes action an a_t chosen from the set of action $\mathcal{A}$ (i.e., move forward by 0.25 m, turn left by 15°, turn right by 15°, and stop). At any time during navigation, if the agent decides to stop and is within 3 meters of the final target, the episode is regarded as successful.

3.1. Overview of Our Approach

In this paper, we use a two-stage framework to solve the VLN-CE task, i.e., the exploration stage and navigation stage, which is similar to the previous work in [4], as the main baseline in this paper. This two-stage setting is more in line with real life in which the agent can build an understanding of the environment through exploration. However, it is natural for this framework to be extended to an online mapping setting. The overview of our proposed framework is shown in Figure 1. In the exploration stage, our agent is allowed to explore the environment to collect the contextual information along the exploration trajectories and build a spatial memory as topological graph G. Then in the navigation stage, given the topological graph G and a language instruction L, the agent is expected to navigate to the goal step-by-step with visual observations. In a two-stage setting, we must consider how and which previously acquired knowledge can be reused in the current environment (i.e., relocation and aggregation problem) and deal with the noise in the knowledge due to the accumulation of the error in the exploration stage. By elaborately designing the structure of ART for this task, we propose the SMART module to effectively acquire and reuse the knowledge. Specifically, the SMART module consists of EM-ART and SM-ART that are modified from fusion ART and will be introduced in Section 3.2. In the navigation stage, our agent uses Landmark Extractor and Node–Landmark Matcher to obtain a waypoint sequence on the topological graph (yellow nodes in Figure 1) in high-level planning (Section 3.3) and executes low-level controlling (Section 3.4) by a navigation model to reach the waypoints sequentially. This kind of modular approach has been proven

to work well in navigation tasks and has significant advantages such as interpretability, robustness, and flexibility [35].

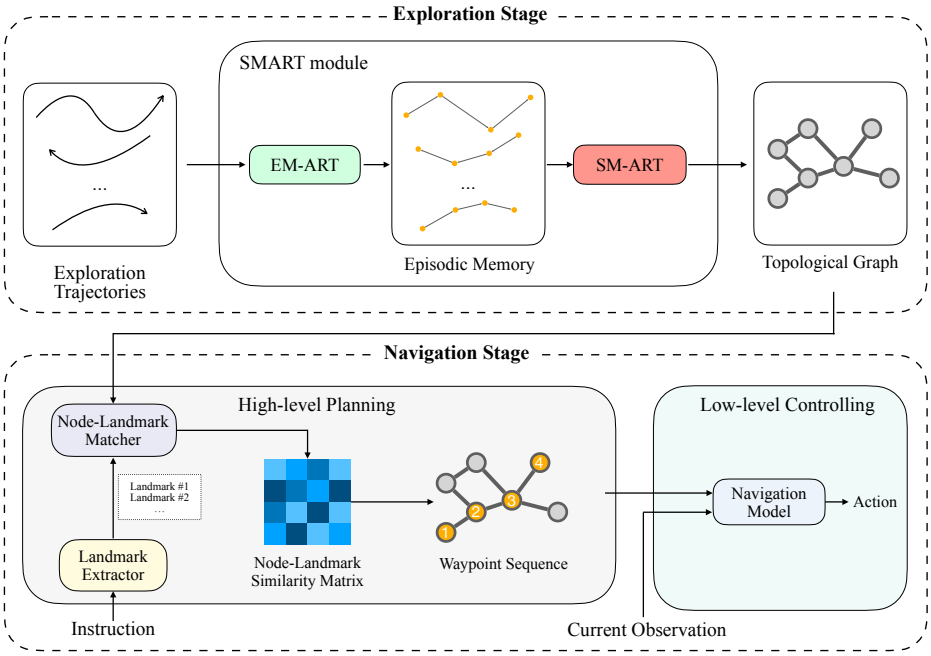


Figure 1. The architecture of our proposed framework. The Self-organizing Memory based on the Adaptive Resonance Theory (SMART) module is employed in exploration stage to build topological graph. During the navigation stage, we decompose the navigation task into two sub-tasks, i.e., the high-level planning and low-level controlling.

3.2. Topological Mapping with Self-Organizing Memory

This section will describe how we build the topological graph using the SMART module in the exploration stage, including fundamentals of the fusion ART theory, episodic memory extraction from exploration trajectories, and spatial memory construction.

ART imitates the information processing of pattern recognition in brains as interactions between subjective expectations and objective sensory inputs [36]. Inspired by this perspective, we use ART as a memory module to explicitly store history information like the environmental layout and visual features in the VLN task, aiming to improve the long-horizon navigational ability of the agent. Before we start to analyze algorithmic details, we introduce multichannel self-organizing fusion ART neural networks which are the foundation blocks. Figure 2 shows the fusion ART model equipped with multiple input fields [13].

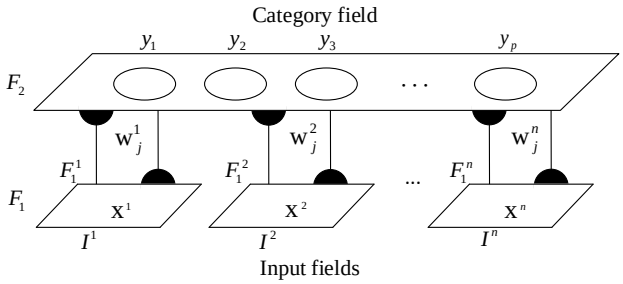


Figure 2. The architecture of fusion ART [13].

3.2.1. Fundamentals of Fusion ART

Here are important terms in fusion ART theory.

1. *Input Vectors:* The input vector and the concatenation of input vectors of all channels can be denoted as $\mathbf{I}^k = (I_1^k, I_2^k, ..., I_{nk}^k)$ and $\mathbf{I} = (\mathbf{I}^1, \mathbf{I}^2, ..., \mathbf{I}^n)$, respectively, where $I_i^k \in [0, 1]$.
2. *Input Fields:* F_1^k is the k-th input field and $\mathbf{x}^k = (x_1^k, x_2^k, ..., x_{nk}^k)$ is the activity vector after receiving $\mathbf{I}^k$.
3. *Category Fields:* F_i $(i > 1)$ means one category field. $\mathbf{y} = (y_1, y_2, ..., y_p)$ is the activity vector of F_2 where p is the number of current learned categories.
4. *Weight Vectors:* $\mathbf{w}_j^k$ is denoted as the weight vector associated with the j-th node in F_2 for learning the input patterns of F_1^k.
5. *Parameters:* The parameters can affect the dynamics of each field, including choice parameters $\alpha^k \geq 0$, learning rate parameters $\beta^k \in [0, 1]$, contribution parameters $\gamma^k \in [0, 1]$, and vigilance parameters $\rho^k \in [0, 1]$.

Then we introduce five crucial steps in fusion ART.

1. The *code activation* process on the j-th node in F_2 is controlled by the choice:

$$T_j = \sum_{k=1}^{n} \gamma^k \frac{\left|\mathbf{x}^k \wedge \mathbf{w}_j^k\right|}{\alpha^k + \left|\mathbf{w}_j^k\right|}. \tag{1}$$

where the fuzzy AND operation $\wedge$ is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i)$, and the norm $|.|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vector $\mathbf{p}$ and $\mathbf{q}$ [13].

2. A *code competition* process selects an F_2 node with the highest T_j indexed at J and sets $y_j = 1$ and $y_j = 0$ for all $j \neq J$.
3. A *template matching* process checks if resonance occurs:

$$m_J^k = \frac{\left|\mathbf{x}^k \wedge \mathbf{w}_J^k\right|}{\left|\mathbf{x}^k\right|} \geq \rho^k, 1 \leq k \leq n. \tag{2}$$

This selection process will be repeated until a category with resonance is found. If no selected node in F_2 meets the vigilance, an uncommitted node is recruited in F_2 as a new category node.

4. A *template learning* process is applied to the connection weights once resonance occurs

$$\mathbf{w}_J^{k(new)} = (1 - \beta^k)\mathbf{w}_J^{k(old)} + \beta^k(\mathbf{x}^k \wedge \mathbf{w}^{k(old)}). \tag{3}$$

5. *Activity Readout*: A chosen F_2 node J may perform a readout of its weight vectors to the input field F_1 such that $\mathbf{x}^{k(new)} = \mathbf{w}_J^k$.

For more details of ART theory, please see this paper [37].

3.2.2. Episodic Memory Extraction from Exploration Trajectories

In the exploration stage, our agent is allowed to move freely in the environment to collect contextual information. Note that autonomous exploration in unknown environments is another challenging problem and has been investigated in prior research [38]. For simplicity, we randomly sample navigable points from the environments and let the agent navigate between each of them. During the exploration process, the EM-ART is activated to extract episodic memory from trajectories. The EM-ART architecture was designed to encode episodic memory from streaming inputs [39] and has been successfully applied in the field of visual navigation [40]. In the following, we will introduce the architecture of EM-ART and the process of episodic memory modeling.

The Architecture of EM-ART. Episodic Memory-ART (EM-ART) is a neural model consisting of two stacked fusion ART networks, providing a method of encoding episodic trajectories in the form of streaming sensory inputs and possible feedback received from the environment [39]. The structure of EM-ART as shown in Figure 3 includes three layers

of memory fields where the lower two layers (F1 and F2) and the upper two layers (F2 and F3) are two different ART modules, respectively. The F1 layer is the attribute layer used to receive from the input patterns and pass to the upper layers the situational attributes. Recognition of an event is triggered by the pattern of activations in F1 in which a cognitive node in F2 is activated as a response to the event. After that, the connection weights between this node on F2 and all F1 fields can be adjusted to determine the activation pattern of the incoming event.

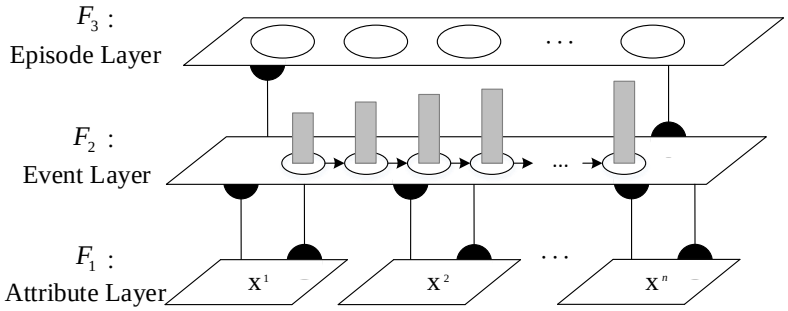


Figure 3. The architecture of EM-ART [39].

The F2 layer is the event layer and serves as a medium-term memory buffer for event activations. The F2 nodes are activated in response to a sequence of events. As activation values in F2 decay over time, a graded pattern of activations represents the sequence's order. For more details on the gradient encoding principle, please see [39].

The F3 layer is the episode layer. Even partial or imprecise clues can recall an existing category by generalization. Using a top-down readout from F3 to F2, the complete episode can be reproduced once a selected node in F3 is recognized. Activations from F2 to F1 can also be used to reproduce events in an episode.

The process of episodic memory modeling. During the process of exploration, our agent collects contextual information alone on the trajectories while learning the data pattern using the EM-ART and generates related episodic memory. Such episodic memory should represent the temporal exploration experience of the agent and implicitly encode the environmental layout by discovering the key locations to enable agents to succeed in navigation. Previous work used the reachability estimator to sparsify the dense exploration trajectories into a topological map [4], but the limitation is that the reachability estimator needs to be trained in the large-scale labeled dataset while EM-ART is a self-organizing method without training. EM-ART takes the list of exploration trajectory $\mathbf{T} = \{T_1, T_2, \ldots, T_m\}$ as input to build each of them into an episode consisting of individual events, where m is the number of exploration trajectories. $T_i = \{O_1, O_2, \ldots, O_n\}$, where n is the total steps of each trajectory and the elements O are the visual embeddings from all observations along the trajectory encoded by the pre-trained Contrastive Language–Image Pre-training (CLIP) [22] image encoder.

Algorithm 1 is the process of episodic modeling. In general, the recognition and learning processes in EM-ART involve bottom-up activation, parallel search in the category layer, top-down template matching, and generalization of connection weights. Not only are these operations invoked when learning events, but also when encoding episodes. To learn the temporal order of events, EM-ART applies gradient encoding when it recognizes an event different from the last sample. The sequential learning method mimics human working memory's invariance principle. By multiplying and decaying their activation values, activated items can retain their temporal order. The learned event nodes are marked as yellow dots in the Episodic Memory rounded rectangle of Figure 1, while the arrows denote the temporal sequence of events within an episode. In the experimental section, we will discuss how to construct the input vectors for the memory module.

Algorithm 1 Episodic memory modeling algorithm

Input: Exploration Trajectories $\{T_1, T_2, ..., T_m\}$
Output: Episodic memory

for all T_i **do**
  for all points in each trajectory **do**
    Get the input vectors O_t
    Normalize the input vectors
    Code activation and competition for F_2 nodes
    if Template matched successful **then**
      Learn weights of an existing event node
    else
      Recruit an uncommitted node
    end if
  end for
  Form an activation vector for selected F_2 nodes
  Learn the connection weights from y to an F_3 node
end for

3.2.3. Spatial Memory Construction

After using the EM-ART to sparsify the exploration trajectories in the environments, we use a three-channel fusion ART, i.e., Spatial Memory-ART (SM-ART) [40] designed to merge the different episodic memory into a consolidated spatial memory. This spatial memory manifests itself as a topological map, including nodes with visual embedding and position information and connectivity relationships between nodes.

The architecture of SM-ART. Firstly, we define the spatial memory as a set of semantic fragments $S = \{S_1, S_2, \ldots, S_\eta\}$, where η is the number of all learned events in EM-ART. In order to identify connections from one node to all other directly reachable nodes, we treat each fragment as a one-to-many relationship. The semantic fragment is defined as a set

$$S_i = \{e_{ij} = (n_i, n_j, h_{ij})\} \tag{4}$$

where each element e_{ij} denotes an edge from n_i to n_j; n_i and n_j are the learned events in EM-ART, and $h_{ij} \in (0, 1)$ is the normalized distance.

SM-ART consists of three input channels as shown in Figure 4 and the input pattern of SM-ART is $\mathbf{I} = (\mathbf{I}^1, \mathbf{I}^2, \mathbf{I}^3)$, where $\mathbf{I}^1$ is the source node, $\mathbf{I}^2$ is the target node and $\mathbf{I}^3$ is the weight of the edge from source node to target node. The dimensions of the input vectors are η. We use one-hot encoding in $\mathbf{I}^1$, i.e., when the i-th event node is the source node of this edge, $I_i^1 = 1$. For other event nodes $j \neq i$, $I_j^1 = 0$. The learning process of SM-ART needs to integrate the input patterns of all edges sent from a certain source node into on category node in the F_2 connectivity layer. We use one-cold encoding in $\mathbf{I}^2$ and $\mathbf{I}^3$ since the fuzzy AND operation $\wedge$ retains the smaller values from the generalized vectors. In $\mathbf{I}^2$, we have $I_j^2 = 0$ where the indicated target node has an event id j. Correspondingly, $I_k^3 = 1$ for all $k \neq j$ and $I_j^3 = h_{ij}$.

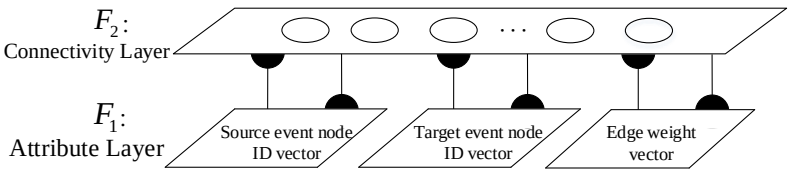


Figure 4. The architecture of SM-ART [40].

The process of spatial memory consolidating. Algorithm 2 describes the process of spatial memory consolidation. When recalling the previously generated episodic memory, each learned episode node in F_3 of EM-ART readouts its connection weights from top to

bottom so that the activity vector at the time of episodic formation reappears in the F_2 layer. The algorithm can recall the temporal order in which landmarks were ever visited by decoding the gradients. With every update to F_1, SM-ART input fields are gradually integrated into a fully qualitative representation of the underlying space through interaction operations between the two layers.

It is possible to discover common contexts between episodes and connect corresponding trajectories by building edges from or to those shared contexts during episodic memory playback. Consequently, events that occur at different times but are actually nearby in space are connected by a few edges through their shared neighboring events, i.e., the discovery of shortcuts. The final output of the SMART module is a directed graph with vertices as visual features and edges encoding traversability and proximity, which will be used as a topological graph by the agent for performing the navigation task.

Algorithm 2 Spatial memory consolidating algorithm

Input: Episodic Memory generated from EM-ART
Output: Learned spatial memory

```
for all Episode node of EM-ART do
    Readout the activation vector to F2
    Recall the related event sequence
    for all Event node of the event sequence do
        Extract the connectivity and build the input pattern
        Update the input field of SM-ART
        Learn by interactions between F1 and F2
    end for
end for
```

3.3. High-Level Planning

After the exploration stage, the agent has constructed the spatial memory of the environment. This kind of spatial memory can be represented as a topological map where the nodes save the visual embedding and edges save the connectivity and weights between nodes. We present a hierarchical VLN framework using a high-level planner for predicting waypoints and a low-level controller for fine-grained actions. In the navigation stage, prior to taking any action steps in the environment, the agent utilizes the topological map and natural language instructions as inputs to execute high-level planning to get a waypoint sequence on the map.

To materialize such an elegant idea, we implement the high-level planning based on two different modalities using two off-the-shelf modules, called Landmark Extractor and Node–Landmark Matcher, respectively. As shown in Figure 5, if the topological map is not discretized enough, the original observation sequence contains a lot of redundancy and noise, which will reduce the efficiency and accuracy of matching. Conversely, if the topological map is too sparse, it will be impossible to distinguish different landmarks in the instruction. We will illustrate this with our experiments in Section 5.1.

3.3.1. Landmark Extractor

Given the natural language instruction L, the Landmark Extractor is expected to extract the sequence of landmark descriptions $\mathcal{L} =< l_1, l_2, \ldots, l_n >$. Recently, transformer architecture models pre-trained on large amounts of Internet text have achieved significant improvements in many NLP tasks and these large-scale pre-trained models can be used directly for downstream tasks without any fine-tuning. Therefore, we chose GPT-3 [41] proposed by OpenAI as the Landmark Extractor since it has shown remarkable zero-shot ability and generalization performance. GPT-3 is a large-scale language model with 175 B parameters trained on 400 B tokens from CommonCrawl data. In zero-shot and few-shot scenarios, the model yields surprisingly effective results. In the experimental section,

we will discuss the effects of using other popular pre-trained models as our Landmark Extractor.

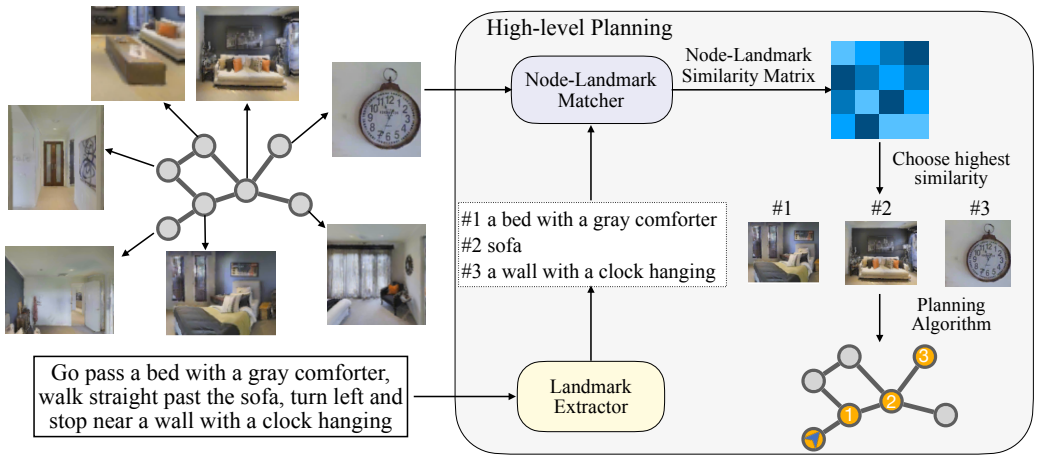


Figure 5. Overview of High-Level Planning.

With only a prompt or conditioning on a few examples, i.e., prompt learning, GPT-3 shows strong performance on a wide variety of tasks without any fine-tuning. Therefore, we need to design the form of input according to the downstream task requirements. For the task of landmark extraction, we encode the task as a part of the text input and design a simple prompt as shown in as following. This prompt contains an example of how to extract landmarks from instructions. When the agent receives the language instruction L, it will be placed in brackets of the prompt, and the whole is passed to the GPT-3 model to obtain landmarks.

> Walk past bottom of stairs. Walk past fireplace. Wait at kitchen desk.
> Landmarks: stairs, fireplace, kitchen desk.
> *{Input Instruction is placed here}*
> Landmarks: ____

3.3.2. Node–Landmark Matcher

The role of the Node–Landmark Matcher is to calculate a cross-modal similarity matrix between the nodes and landmarks as shown in Figure 1, where the nodes V are from the topological graph G and the landmarks $\mathcal{L}$ are the output of the Landmark Extractor. Through the similarity matrix, we can find the probability $P(v_i|l_j)$ that node v_i refers to landmark description l_j and chooses the node with the highest probability as the matched node. This task can be regarded as an image–text retrieval task because landmark descriptions are in text and the features of nodes are image embeddings. CLIP is a pre-trained model that jointly encodes images and texts into a common embedding space, which enables a more effective comparison of the similarity between two modalities. We use CLIP as the Node–Landmark Matcher since it has a robust zero-shot ability for image-text retrieval tasks.

In our framework, we encode the landmarks $\mathcal{L}$ through the CLIP text encoder to obtain language embeddings. During the exploration stage, the agent encodes the visual observation along with the exploration trajectory using the CLIP image encoder and constructs spatial memory about the scene using the SMART module. When performing high-level planning, it can extract stored image embeddings and input them into the Node–Landmark Matcher. For improving the matching performance, we use a simple prompt added to the landmark, in the form of *"A photo of [landmark]"*, where the landmark in brackets will be replaced by the extracted landmarks $\mathcal{L}$. Through the output matrix of the Node–Landmark Matcher, we can find a node sequence $\mathcal{V} = < v_1, v_2, \ldots, v_n >$ corresponding to landmarks

$\mathcal{L}$. Then we use a planning algorithm to find a path in the topological graph that contains all nodes in $\mathcal{V}$ and denote the path as $\mathcal{P} = < w_1, w_2, \ldots, w_\epsilon >$, where ϵ is the number of path nodes. In general, the number of path nodes ϵ is greater than or equal to the length of matched node sequences $\mathcal{V}$ since the traversal may also cover other nodes in between those matched ones. The planned path $\mathcal{P}$ consists of the image features and positions of all nodes and will be executed in low-level controlling.

3.4. Low-Level Controlling

After generating a path in high-level planning, this path will be translated into robot actions in low-level controlling. Previous work [4] uses a very simple low-level controller to orient the agent towards the waypoint and then move in a straight line. If there are obstacles on the path, this method is difficult to control the agent to reach the goal successfully. Instead, we use an off-the-shelf deep reinforcement learning model, named decentralized distributed proximal policy optimization (DD-PPO) [42] to navigate the nodes on the path. Since spatial memory is preserved during the exploration stage, the agent can recall the image features and position information of all nodes in $\mathcal{P}$. Therefore, the execution process of this plan can be decomposed into a series of point goal navigation tasks. Using the current RGB-D observation O_t and the planned path $\mathcal{P}$, the DD-PPO model can output the low-level actions for the agent and reach the target position finally. Please refer [42] to more detail of the DD-PPO method.

4. Experiments

4.1. Dataset and Evaluation Metrics

The experiments were conducted in the VLN-CE [8] dataset based on the Habitat Simulator [15], which contains 16,844 path-instruction pairs over 90 scenes. We used the standard metrics to evaluate our agents as described in paper [1]: Success Rate (SR), Success weighted by Path Length (SPL), Oracle Success Rate (OSR), Trajectory Length (TL), Navigation Error (NE), and normalized Dynamic Time Warping (nDTW). Please see [1,43] for details of the metrics. We implemented our agent using PyTorch and the Habitat simulator v0.1.7. We built our code on top of the VLN-CE codebase (https://github.com/jacobkrantz/VLN-CE, accessed on 1 September 2023) and use the same set of hyperparameters.

4.2. Model Summary

The six strong baseline models for the VLN-CE task are compared with our proposed approach. According to whether topological graphs are used in these models, we categorize them as topology-based models and non-topology-based models.

Non -topology-based models:

- Seq2Seq [8]: Proposes an end-to-end model that uses the LSTM network to predict actions directly from a language instruction and images.
- CMA+PM+DA+Aug [8]: Proposes a cross-modal attention model to grounding the language instruction to visual observation, combined with Progress Monitor techniques, imitation policy DAgger, and the Speaker–Follower data augmentation method.
- SARSA [44]: Proposes a hybrid transformer-recurrence model that focuses on combining classical semantic mapping techniques with a learning-based method.
- LAW [45]: Proposes a new supervision signal, which utilizes the nearest waypoint on the path as the supervision signal instead of the target position.
- Waypoint Model [46]: Proposes an end-to-end model that predicts relative waypoints directly from natural language instructions and panoramic RGB-D inputs.

Topology-based models:

- CMTP [4]: Applies the most similar framework to ours among the baselines, which prebuilds a topological graph of the spatial layout. However, it uses a more complicated

architecture for navigation planning, i.e,. a cross-modal transformer, which requires large-scale training. The authors have not released their source code, so we directly report the results shown in the paper.

4.3. Research Questions

We focus on answering the following research questions to guide our experiments:
RQ1: What are the effects of different ART parameters on the composition of the topological map?
RQ2: What is the effect of using different pre-trained models in the SMART framework on navigation performance?
RQ3: Does SMART improve overall navigation performance compared to the baselines for the VLN-CE task?

5. Results and Discussion

5.1. Results of Graph Construction

To answer **RQ1**, we investigate the results of graph construction from two aspects: different input patterns in EM-ART and different vigilance parameters ρ, which are two main factors that affect the recognition and generalization ability of EM-ART. Different input patterns also lead to significant differences in what EM-ART learns. The higher the value of ρ, the higher the threshold of similarity between nodes, and the denser the whole topological graph will be. In Section 5.1.1 we design two different EM-ART input patterns and conduct some comparative experiments to find out the most suitable pattern under the VLN-CE environment. In Section 5.1.2 we will discuss the effect of the vigilance parameter on building topological graphs.

5.1.1. Effect of Different Input Patterns

Two different input patterns for EM-ART are designed. One pattern is introduced in Section 3.2.2 using partial visual observation embeddings $O_{partial}$ (with a horizontal field-of-view of 90°) and the agent orientations d as input vector, i.e., $\mathbf{I}_{partial} = [O_{partial}, d]$. This pattern can use orientation information to distinguish similar observations at different locations, thereby alleviating the problem of perceptual aliasing. Considering that the panoramic image contains enough information about the surrounding environment, We design another pattern $\mathbf{I}_{pano} = [O_{pano}]$ that uses the panoramic observation embeddings O_{pano} as only the input vector. The demonstration of two patterns is shown in Figure 6. We evaluate the graph construction results of these two different input patterns to find out which pattern is more suitable for the VLN-CE environment.

Figure 6. Comparison of two input patterns. (**a**) Partial view + orientation, (**b**) panoramic view.

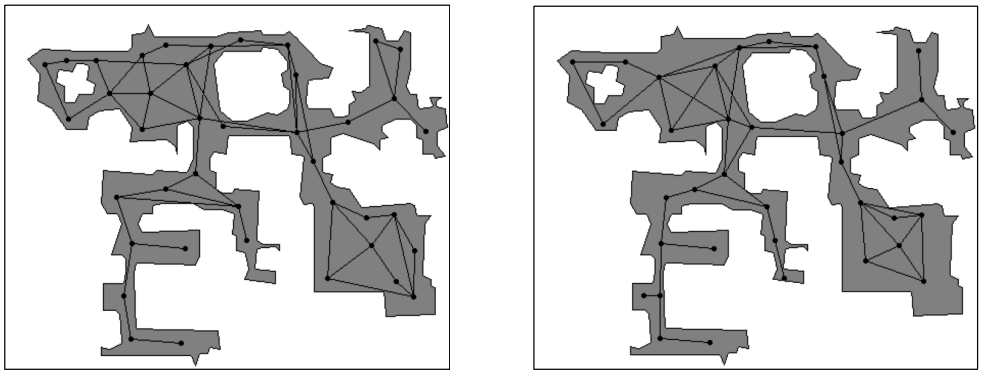
These two different patterns are employed to build topological graphs for 90 VLN-CE scenarios respectively. Table 1 provides the statistics of topological graphs based on the two pattern designs and two previous works [1,47], where the numbers of nodes and edges, and edge lengths are compared. It can be found that the total number of nodes in the topological graph of MP3D is the smallest, and the graphs constructed by $\mathbf{I}_{partial}$ contain the largest number of nodes and edges. In contrast, the graphs constructed by $\mathbf{I}_{pano}$ are relatively sparse, with fewer nodes, and the length of each edge is the longest

among all the graphs, indicating a better sparsity effect. The MP3D graph is directly migrated from the discrete environment, and there is no corresponding adaptation to the continuous environment. Consequently, there are some problems, such as the position of some nodes in the unnavigable area, and the connection relationship of some edges is also blocked by obstacles. The graphs constructed by Hong2022 [47] are adapted from the MP3D graphs for the continuous environment by utilizing some handcrafted rules. However, this approach does not form an understanding of the environment, nor does the location of nodes necessarily lie in critical locations. It only has the connectivity relationship between nodes, so it is very limited in helping navigation tasks.

Table 1. A comparison of different topological graph statistics.

	Overall Nodes	Number of Edges per Scene	Number of Nodes per Scene	Average Edge Length	Average Edges of Each Node
MP3D [1]	10,559	218.7	117.3	1.87	4.07
Hong2022 [47]	13,358	245.9	148.4	2.26	3.31
$\mathbf{I}_{pano}$	12,150	226.5	135.0	2.35	3.15
$\mathbf{I}_{partial}$	14,287	258.4	158.7	1.75	4.22

In order to compare and analyze the effect between the two EM-ART input patterns, we select results of topological graphs constructed by two different patterns in a specific scene, as demonstrated in Figure 7. Obviously, both patterns do a good job of extracting the structure of the environment, but $\mathbf{I}_{partial}$ has more nodes, resulting in some redundancy. In comparison, the graphs constructed by $\mathbf{I}_{pano}$ can generate nodes at each key node position. Based on this observation, we finally use $\mathbf{I}_{pano}$ as the input pattern of EM-ART to generate spatial memory.



(**a**) (**b**)

Figure 7. Examples of graph construction results. (**a**) Graph constructed by $\mathbf{I}_{partial}$, (**b**) Graph constructed by $\mathbf{I}_{pano}$.

5.1.2. Effect of Different Vigilance Parameters

In order to investigate the effect of different vigilance parameters on the results of graph construction, we select 10 scenes, each with 3 navigation episodes, to perform the VLN-CE task using the constructed topological graphs. The setting in this part of the experiments includes the total steps in the exploration stage in each scene of about 500, the contribution parameters $\gamma = 0.0001$, and learning rate parameters $\beta = 0.4$.

Figure 8 shows the effect of ρ from 0.7–1 on the number of event nodes and the corresponding navigation success rate. The results in the figure are the average of 10 scenes. It can be seen that when ρ is in the range of 0.7–0.9, the number of event nodes is smaller than 10. The vigilance parameter represents the similarity threshold between the embeddings of different visual observations, but most visual observations in indoor scenes are relatively

similar. The smaller threshold will cause more nodes to be considered similar and very limited nodes to be saved. However, it is difficult to represent a scene structure with very limited event nodes. When $\rho = 1.0$, all existing event nodes cannot satisfy top-down matching with a novel input unless the input is exactly the same as one of the existing event nodes. In this situation, all observations of the exploration path are basically preserved, so the final number of event nodes is close to the number of exploration steps.

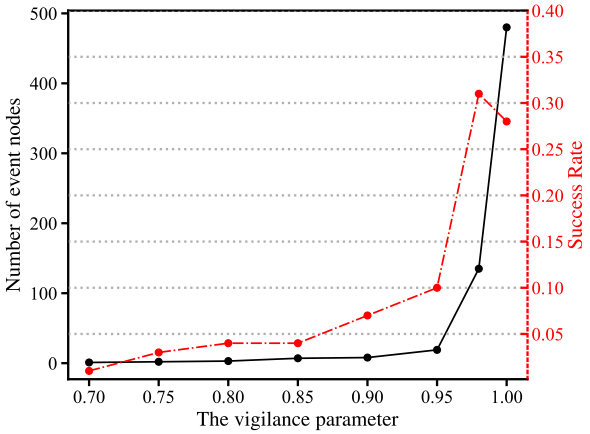


Figure 8. Effect of vigilance parameter on event clustering and the performance of navigation task.

Using the topological graph constructed under different vigilance parameters, we use our full system to perform 30 navigation episodes in 10 scenes. The Success Rate is used as an evaluation metric to measure the performance of the system. Figure 8 demonstrates that when the ρ is between 0.7–0.9, the navigation task fails most of the time. In this case, too few nodes have made it difficult for the agent to match the landmark accurately. Even if the match is successful, it is not guaranteed that the node must be within 3 m of the target location. When the ρ is between 0.9–0.1, the success rate of navigation is improved significantly and peaks at $\rho = 0.98$. However, the success rate does not increase linearly with the increase of ρ because the success rate decreases to a certain extent when $\rho = 0.98$. The main reason for this is that the very large ρ reduces the scene generalization ability of the SMART module. Additionally, too many nodes might affect the matching result between the nodes and landmarks, which leads to a reduction in the success rate. In conclusion, we set $\rho = 0.98$ as the default value for the following experiments.

5.2. Results of Landmark Extraction

The Landmark Extractor is an important part of our framework. The quality of extracted landmarks directly affects the alignment between visual scenes and instructions, and the final navigation effect. A reliable Landmark Extractor should be able to extract landmarks from natural language instructions in order. For the landmark extraction experiment, we use different pre-trained large language models as the Landmark Extractor to find the performance difference. Specifically, we compare the fairseq-125M [48], fairseq-13B [48], GPT-J-6B [49], GPT-NeoX-20B [50], and GPT-3 [41] and the non-learning method. This non-learning method uses the spaCy [51] NLP library to extract the base noun phrases and then removes some manually pre-defined words. For parsing simple queries, we designed a simple prompt to improve the in-context learning ability of the pre-trained models. We manually annotated 50 pairs of data (instruction, landmark sequence) as human supervision for model performance evaluation. For a comprehensive evaluation, we designed the Extraction Success Rate (ESR) metric as a score for extraction results:

$$ESR = \frac{|\phi(\mathcal{L}, \hat{L})|}{|\hat{L}|}, \tag{5}$$

where the $\mathcal{L}$ is the list of extracted landmarks by the model and $\hat{L}$ is the list of ground-truth landmarks while $\phi(\cdot)$ is a function to calculate the longest common subsequence and $|\cdot|$ means to calculate the length of the list. By using this metric, we are able to determine not only whether the landmark sequences are extracted correctly, but also whether they correspond to the ground truth sequence. As shown in Table 2, GPT-3 significantly outperforms other models, owing to its superior representation capabilities and in-context learning.

Table 2. Landmark Extractor performance.

Model Name	Extraction Success Rate
fairseq-125M	0.105
GPT-NeoX-20B	0.271
fairseq-13B	0.323
GPT-J-6B	0.474
spaCy	0.704
GPT-3 (text-davinci-002)	0.837

5.3. Evaluation on VLN-CE

To answer **RQ3**, we investigate the performance of our full system and other baseline models, as presented in Table 3.

Table 3. Comparison on agent performance in VLN-CE val-seen and test dataset. Note that all these baseline models are trained on large-scale supervised datasets, whereas our framework uses off-the-shelf models without fine-tuning. The top two results are in bold and underlined, respectively.

Models	Val-Unseen						Test				
	TL↓	NE↓	nDTW↑	OSR↑	SR↑	SPL↑	TL↓	NE↓	OSR↑	SR↑	SPL↑
Non-topology-based Methods											
Seq2Seq [8]	9.32	7.77	–	37	25	22	<u>8.13</u>	7.85	27	20	18
CMA+PM+DA+Aug [8]	8.64	7.37	51	40	32	30	8.85	7.91	36	28	25
SASRA [44]	<u>7.89</u>	8.32	47	–	24	22	–	–	–	–	–
Waypoint Model [46]	**7.62**	**6.31**	–	40	**36**	**34**	**8.02**	<u>6.65</u>	<u>37</u>	**32**	**30**
LAW [45]	8.89	6.83	**54**	**44**	35	31	–	–	–	–	–
Topology-based Methods											
CMTP [46]	–	7.90	–	39	23	19	–	–	–	–	–
Ours (SMART)	10.90	<u>6.35</u>	<u>52</u>	<u>42</u>	<u>31</u>	<u>27</u>	9.85	**6.62**	**40**	**32**	<u>28</u>

Table 3 shows our final results on VLN-CE, where the test result is from the Leaderboard website: https://eval.ai/web/challenges/challenge-page/719/leaderboard/1966, accessed on 1 September 2023. It is worth noting that the SMART model proposed in this paper does not need any annotated data for training, namely zero-shot. In contrast, previous works required training on the training data and then testing on the test dataset. This kind of supervised learning decreases the performance a great deal when tested in the previous unseen scenes. Comparatively, the method proposed in this paper shows consistent performance on both val-unseen and test datasets, which illustrates the effectiveness of this method. The CMTP method, similarly to this work, also explores the environment first to build topological graphs and then uses the graphs in the navigation stage for planning and navigation. According to the results, the performance of our model on SR and SPL surpassed the CMTP method by a large margin, i.e., 31% versus 23% in SR and 27% versus 19% in SPL, respectively. In terms of test set, our approach even surpassed some supervised methods, such as Seq2Seq and CMA models, and is comparable with the Waypoint Model (32% in SR).

Figure 9 shows the qualitative results of our full system. The top of the figure is the instruction while the underlined text is the extracted landmark and the number in the upper right corner of the landmark represents the sequence order. The panoramic observation with the number on the right side of the figure is the visual observation of the nodes

matched by the corresponding landmarks. We mark the corresponding landmarks with red boxes in the panoramic observations. The figure also contains the planned waypoint sequence, the ground-truth path, and the path actually traveled by the agent. It can be seen from the figure that our model can extract key landmarks, e.g., the couches and the bed, and make correct matches, followed by planning a reasonable path to visit these nodes in turn and finally reach the target position. This modular framework makes it easy to interpret the agent's purpose and evaluate related abilities.

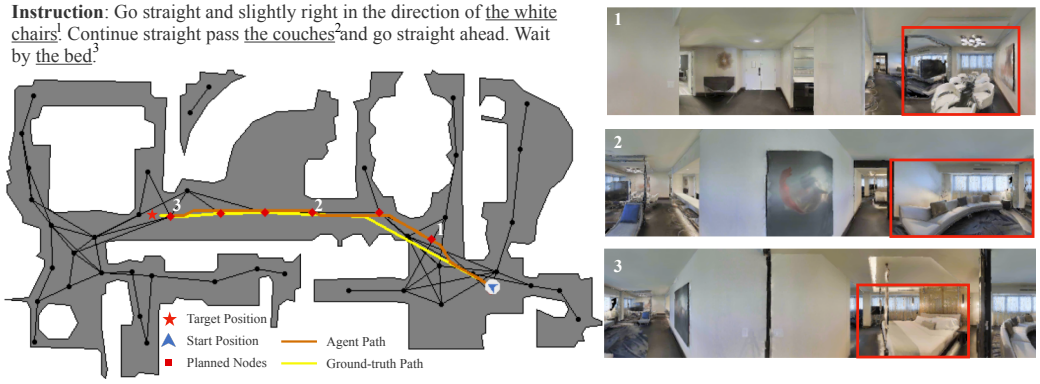


Figure 9. Qualitative example of the navigation episode.

5.4. Failure Analysis

By delving into more experiments, we find that the main reasons why the agent cannot complete the navigation task in some cases are as follows:

- There are no obvious landmarks in some language instructions. For example, *"Turn around, move forward 4 steps, stop"* is an instruction from the training dataset. In this situation, the Landmark Extractor cannot extract any landmarks, so the episode fails directly.
- From the perspective of image-text matching, it may be walls or doors that occupy most of the area on the indoor panoramic image, but some salient landmarks may only occupy a small area on the image, which will greatly affect matching performance.
- The characteristics of the indoor environment might be one of the reasons for navigation failure. There could be many similar landmarks in the indoor scene, e.g., several beds or tables in the same house. During the node–landmark matching process, the agent struggles to distinguish which table or bed it corresponds to.

6. Conclusions

Inspired by psychological findings, we proposed the SMART module based on ART to address topological mapping and a simple and modular framework incorporating three off-the-shelf pre-trained models to execute the VLN-CE task. The topological graphs are formed by aggregating the spatial–temporal contexts around landmarks. By decomposing VLN-CE tasks, each pre-trained large model can be used directly in our framework without training for extracting landmark descriptions from instructions, matching them with visual scenes and controlling low-level actions. These modules work in a hierarchical manner including high-level planning for waypoints and low-level movement between waypoints. The empirical results show that the proposed model can construct topological maps effectively and can perform competitively on the VLN-CE tasks, outperforming some supervised models and showing excellent generalization ability.

For future direction, improving the performance of pre-trained models is the most direct way to improve the success rate. The framework cannot handle the situation where the instruction does not have a salient landmark, so we need to think about how to modify this framework. In addition, adding scene graphs with more semantic information to the features of nodes to make nodes more discriminative is also a promising direction in the future.

Author Contributions: Conceptualization, W.W.; Methodology, W.W. and Y.H.; Software, K.X.; Validation, W.W.; Formal analysis, L.Q.; Resources, K.X.; Data curation, K.X.; Writing—original draft, W.W.; Writing—review & editing, Y.H.; Visualization, L.Q.; Supervision, Q.Y.; Funding acquisition, Q.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China: 62103420, 62103425, 62103428, 62306329 and Natural Science Fund of Hunan Province: 2023JJ40676, 2021JJ40697, 2021JJ40702.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Anderson, P.; Wu, Q.; Teney, D.; Bruce, J.; Johnson, M.; Sünderhauf, N.; Reid, I.; Gould, S.; Van Den Hengel, A. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3674–3683.
2. Deng, Z.; Narasimhan, K.; Russakovsky, O. Evolving graphical planner: Contextual global planning for vision-and-language navigation. *arXiv* **2020**, arXiv:2007.05655.
3. Zhu, F.; Liang, X.; Zhu, Y.; Yu, Q.; Chang, X.; Liang, X. SOON: Scenario Oriented Object Navigation with Graph-based Exploration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtually, 19–25 June 2021; pp. 12689–12699.
4. Chen, K.; Chen, J.K.; Chuang, J.; Vázquez, M.; Savarese, S. Topological Planning with Transformers for Vision-and-Language Navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 11276–11286.
5. Chen, S.; Guhur, P.L.; Schmid, C.; Laptev, I. History Aware Multimodal Transformer for Vision-and-Language Navigation. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 5834–5847.
6. Hong, Y.; Wu, Q.; Qi, Y.; Opazo, C.R.; Gould, S. Vln bert: A recurrent vision-and-language bert for navigation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 1643–1653.
7. Hao, W.; Li, C.; Li, X.; Carin, L.; Gao, J. Towards learning a generic agent for vision-and-language navigation via pre-training. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 13137–13146.
8. Krantz, J.; Wijmans, E.; Majumdar, A.; Batra, D.; Lee, S. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 104–120.
9. Gillner, S.; Mallot, H.A. Navigation and acquisition of spatial knowledge in a virtual maze. *J. Cogn. Neurosci.* **1998**, *10*, 445–463. [CrossRef]
10. Foo, P.; Warren, W.H.; Duchon, A.; Tarr, M.J. Do humans integrate routes into a cognitive map? Map-versus landmark-based navigation of novel shortcuts. *J. Exp. Psychol. Learn. Mem. Cogn.* **2005**, *31*, 195. [CrossRef] [PubMed]
11. Lin, C.; Jiang, Y.; Cai, J.; Qu, L.; Haffari, G.; Yuan, Z. Multimodal Transformer with Variable-length Memory for Vision-and-Language Navigation. *arXiv* **2021**, arXiv:2111.05759.
12. Janzen, G.; Van Turennout, M. Selective neural representation of objects relevant for navigation. *Nat. Neurosci.* **2004**, *7*, 673–677. [CrossRef]
13. Tan, A.H.; Carpenter, G.A.; Grossberg, S. Intelligence through interaction: Towards a unified theory for learning. In Proceedings of the International Symposium on Neural Networks, Nanjing, China, 3–7 June 2007; pp. 1094–1103.
14. Tan, A.H.; Subagdja, B.; Wang, D.; Meng, L. Self-organizing neural networks for universal learning and multimodal memory encoding. *Neural Netw.* **2019**, *120*, 58–73. [CrossRef]
15. Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; et al. Habitat: A Platform for Embodied AI Research. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
16. Wu, W.; Chang, T.; Li, X. Vision-and-language navigation: A survey and taxonomy. *arXiv* **2021**, arXiv:2108.11544.
17. Wang, H.; Wang, W.; Liang, W.; Xiong, C.; Shen, J. Structured Scene Memory for Vision-Language Navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8455–8464.
18. Zhu, F.; Zhu, Y.; Chang, X.; Liang, X. Vision-Language Navigation With Self-Supervised Auxiliary Reasoning Tasks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
19. Ma, C.Y.; Wu, Z.; AlRegib, G.; Xiong, C.; Kira, Z. The regretful agent: Heuristic-aided navigation through progress estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 6732–6740.

20. Wang, X.; Huang, Q.; Celikyilmaz, A.; Gao, J.; Shen, D.; Wang, Y.F.; Wang, W.Y.; Zhang, L. 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, Long Beach, CA, USA, 15–20 June 2019; pp. 6629–6638.
21. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv* **2019**, arXiv:1910.03771.
22. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 8748–8763.
23. Tolman, E.C. Cognitive maps in rats and men. *Psychol. Rev.* **1948**, *55*, 189. [CrossRef] [PubMed]
24. Burgess, N.; Maguire, E.A.; O'Keefe, J. The human hippocampus and spatial and episodic memory. *Neuron* **2002**, *35*, 625–641. [CrossRef] [PubMed]
25. Götze, J.; Boye, J. Learning landmark salience models from users' route instructions. *J. Locat. Based Serv.* **2016**, *10*, 47–63. [CrossRef]
26. Savinov, N.; Dosovitskiy, A.; Koltun, V. Semi-parametric topological memory for navigation. *arXiv* **2018**, arXiv:1803.00653.
27. Caduff, D.; Timpf, S. On the assessment of landmark salience for human navigation. *Cogn. Process.* **2008**, *9*, 249–267. [CrossRef] [PubMed]
28. Gupta, S.; Fouhey, D.; Levine, S.; Malik, J. Unifying map and landmark based representations for visual navigation. *arXiv* **2017**, arXiv:1712.08125.
29. Hong, Y.; Rodriguez, C.; Qi, Y.; Wu, Q.; Gould, S. Language and visual entity relationship graph for agent navigation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7685–7696.
30. Merity, S.; Keskar, N.S.; Socher, R. Regularizing and optimizing LSTM language models. *arXiv* **2017**, arXiv:1708.02182.
31. Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; et al. Learning to Navigate in Complex Environments. In Proceedings of the 5th International Conference on Learning Representations, ICLR, Toulon, France, 24–26 April 2017.
32. Anderson, P.; Shrivastava, A.; Parikh, D.; Batra, D.; Lee, S. Chasing ghosts: Instruction following as bayesian state tracking. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 369–379.
33. Gupta, S.; Davidson, J.; Levine, S.; Sukthankar, R.; Malik, J. Cognitive mapping and planning for visual navigation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2616–2625.
34. Chaplot, D.S.; Salakhutdinov, R.; Gupta, A.; Gupta, S. Neural topological slam for visual navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 12875–12884.
35. Karkus, P.; Ma, X.; Hsu, D.; Kaelbling, L.P.; Lee, W.S.; Lozano-Pérez, T. Differentiable algorithm networks for composable robot learning. *arXiv* **2019**, arXiv:1905.11602.
36. Carpenter, G.A.; Grossberg, S. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comput. Vision Graph. Image Process.* **1987**, *37*, 54–115. [CrossRef]
37. Wang, W.; Tan, A.H.; Teow, L.N. Semantic memory modeling and memory interaction in learning agents. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *47*, 2882–2895. [CrossRef]
38. Chaplot, D.S.; Gandhi, D.; Gupta, S.; Gupta, A.; Salakhutdinov, R. Learning To Explore Using Active Neural SLAM. In Proceedings of the 8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, 26–30 April 2020.
39. Wang, W.; Subagdja, B.; Tan, A.H.; Starzyk, J.A. Neural modeling of episodic memory: Encoding, retrieval, and forgetting. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 1574–1586. [CrossRef]
40. Hu, Y.; Subagdja, B.; Tan, A.H.; Yin, Q. Vision-Based Topological Mapping and Navigation with Self-Organizing Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 7101–7113. [CrossRef]
41. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
42. Wijmans, E.; Kadian, A.; Morcos, A.; Lee, S.; Essa, I.; Parikh, D.; Savva, M.; Batra, D. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv* **2019**, arXiv:1911.00357.
43. Anderson, P.; Chang, A.; Chaplot, D.S.; Dosovitskiy, A.; Gupta, S.; Koltun, V.; Kosecka, J.; Malik, J.; Mottaghi, R.; Savva, M.; et al. On evaluation of embodied navigation agents. *arXiv* **2018**, arXiv:1807.06757.
44. Irshad, M.Z.; Mithun, N.C.; Seymour, Z.; Chiu, H.P.; Samarasekera, S.; Kumar, R. Sasra: Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. *arXiv* **2021**, arXiv:2108.11945.
45. Raychaudhuri, S.; Wani, S.; Patel, S.; Jain, U.; Chang, A.X. Language-aligned waypoint (law) supervision for vision-and-language navigation in continuous environments. *arXiv* **2021**, arXiv:2109.15207.
46. Krantz, J.; Gokaslan, A.; Batra, D.; Lee, S.; Maksymets, O. Waypoint models for instruction-guided navigation in continuous environments. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 15162–15171.
47. Hong, Y.; Wang, Z.; Wu, Q.; Gould, S. Bridging the Gap Between Learning in Discrete and Continuous Environments for Vision-and-Language Navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 15439–15449.

48. Artetxe, M.; Bhosale, S.; Goyal, N.; Mihaylov, T.; Ott, M.; Shleifer, S.; Lin, X.V.; Du, J.; Iyer, S.; Pasunuru, R.; et al. Efficient large scale language modeling with mixtures of experts. *arXiv* **2021**, arXiv:2112.10684.
49. Wang, B.; Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. May 2021. Available online: https://github.com/kingoflolz/mesh-transformer-jax (accessed on 1 September 2023).
50. Black, S.; Biderman, S.; Hallahan, E.; Anthony, Q.; Gao, L.; Golding, L.; He, H.; Leahy, C.; McDonell, K.; Phang, J.; et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv* **2022**, arXiv:2204.06745.
51. Honnibal, M.; Montani, I.; Van Landeghem, S.; Boyd, A. spaCy: Industrial-Strength Natural Language Processing in Python. 2020. Available online: https://zenodo.org/record/8409320 (accessed on 1 September 2023).