Action-Aware Visual-Textual Alignment for Long-Instruction Vision-and-Language Navigation

BOWEN HUANG, School of Computer Science and Technology, Shandong University, China
YANWEI ZHENG*, School of Computer Science and Technology, Shandong University, China
CHUANLIN LAN, Department of Electrical Engineering, City University of Hong Kong, China
DONGCHEN SUI, School of Computer Science and Technology, Shandong University, China
XINPENG ZHAO, School of Computer Science and Technology, Shandong University, China
XIAO ZHANG, School of Computer Science and Technology, Shandong University, China
MENGBAI XIAO, School of Computer Science and Technology, Shandong University, China
DONGXIAO YU, School of Computer Science and Technology, Shandong University, China

Traditional vision-and-language navigation (VLN) requires an agent to navigate to a target location solely based on visual observations, guided by natural language instructions. Compared to this task, long-instruction VLN involves longer instructions, extended trajectories, and the need to consider more contextual information for global path planning. As a result, it is more challenging and requires accurately aligning the instructions with the agent's current visual observations, which is accompanied by two significant issues. Firstly, there is a misalignment between actions. The visual observations of the agent at each step lack explicit action-related details, while the instructions contain action-oriented words. Secondly, there is a misalignment between global instructions and local visual observations. The instructions describe the entire navigation trajectory, whereas the agent's visual observations only provide localized information about a specific position along the trajectory. To address these issues, this paper introduces the Action-Perception Alignment Framework (APAF). In this framework, we first design the Action-Contextual Encoding Module (ACEM), which enriches the agent's visual perception by encoding potential actions with relative heading and elevation angles. We then propose the Dynamic Instruction Weighting Module (DIWM), which adjusts the importance of instruction words based on the agent's current visual observations, emphasizing those words most relevant to the agent's visual observations. Our approach significantly outperforms existing methods, achieving state-of-the-art results with improvements of 8.5% and 4.0% in Success Rate (SR) on the long-instruction R4R and RxR datasets, respectively.

CCS Concepts: • **Computing methodologies** → **Vision for robotics**.

Additional Key Words and Phrases: Long-Instruction Vision-and-Language Navigation, Action-Perception Alignment Framework, Action-Contextual Encoding Module, Dynamic Instruction Weighting Module

*Corresponding author.

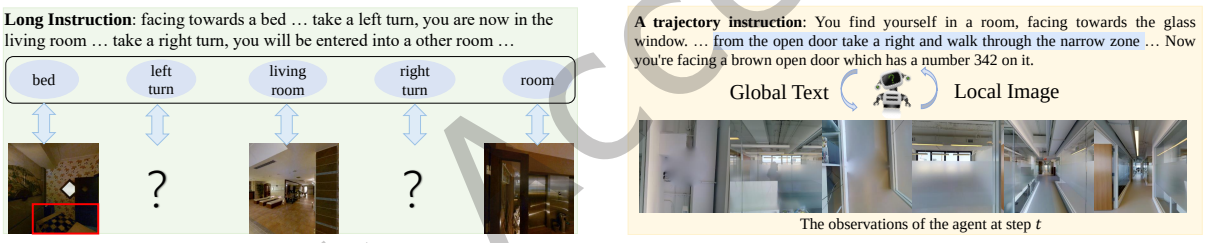
This work was supported by the Natural Science Foundation of Shandong Province, China (Grant No. ZR2022ZD02).
Our code and trained models are available at https://github.com/visee-sdu/APAF.
Authors' addresses: B. Huang, Y. Zheng, D. Sui, X. Zhao, X. Zhang, M. Xiao, and D. Yu, School of Computer Science and Technology, Shandong University, Qingdao, China; emails: huangbw@mail.sdu.edu.cn, zhengyw@sdu.edu.cn, suidongchen@mail.sdu.edu.cn, zhaoxp1001@gmail.com, xiaozhang@sdu.edu.cn, xiaomb@sdu.edu.cn, dxyu@sdu.edu.cn; C. Lan, Department of Electrical Engineering, City University of Hong Kong, Hong Kong, China; email: cllan2-c@my.cityu.edu.hk.

ACM 1551-6865/2025/7-ART
https://doi.org/10.1145/3748656

1 INTRODUCTION

As a task that combines vision, language, and decision-making, vision-and-language navigation (VLN) involves guiding autonomous agents to navigate in unseen environments based on natural language instructions and current visual observations [7, 24, 29, 75]. In early research [21, 27, 41, 49], these instructions were relatively straightforward and brief, requiring only limited movement from the agents. These studies are inadequate for more complex scenarios, such as navigating large indoor environments with multiple rooms or areas to pass through [36]. To address this limitation, recent works have introduced the long-instruction VLN task [11, 32], which requires agents to follow more complex, multi-step instructions that guide them through longer and more complex paths.

Existing research has made significant achievements in the field of VLN. Early methods [21, 41, 49, 63] typically used RNN models or transformer architectures to store various visual information. For example, VLN-BERT [27] incorporates a recurrent function to maintain cross-modal state information, overcoming challenges in adapting BERT to decision-making process of VLN. ADAPT [41] uses CLIP for cross-modality alignment and introduces a modality alignment loss and sequential consistency loss, enabling better action prediction and sequential navigation. Recent studies [2, 13, 40, 70] have focused on integrating more features to enhance decision-making accuracy. For instance, HAMT [11] uses a hierarchical vision transformer to integrate instruction history and current observations. DNA [7] incorporates direction clues from instructions to improve navigation accuracy. However, these studies are primarily designed for traditional VLN task and struggle to accurately align visual observations with long instructions. This makes it difficult to make correct decisions when directly applied to long-instruction VLN.



(a) The absence of action information in visual observations results in challenges in aligning visual and linguistic data.

(b) Global instructions and local visual observations lead to difficulties in aligning text and images

Fig. 1. The difficulties of aligning visual and textual information in the long-instruction scene

In long-instruction VLN, the agent must understand longer instructions and leverage more contextual information for global path planning. In this process, how to align the agent's visual observations with the extended instructions is a significant challenge, which involves two key issues. Firstly, there is a misalignment between actions. As illustrated in Fig. 1a, textual instructions typically involve references to rooms, objects, and actions. However, the visual observations predominantly capture only room and object features, with action-related features often underutilized or absent. This gap results in an incomplete modality alignment, as the critical action-related context present in the instructions is not reflected in the visual input. Secondly, there is a misalignment between global instructions and local visual observations. As demonstrated in Fig. 1b, while the instruction provides a comprehensive trajectory from start to destination, the agent's visual perception is inherently localized, capturing only a snapshot of the environment at each step. This discrepancy between the global instruction and the agent's limited, local visual information further complicates the alignment between these two modalities.

To address these issues of long-instruction VLN, we propose the Action-Perception Alignment Framework (APAF). In this framework, we first design the Action-Contextual Encoding Module (ACEM). It enhances the agent's visual perception by incorporating potential action information and encoding a range of possible actions (e.g., turning left, moving forward) into a feature space. These action features are then integrated with the visual data, providing a richer and more comprehensive representation of the environment. This fusion ensures that both visual information and action-related components of the instructions are better aligned, supporting more informed decision-making at each step. Then we propose the Dynamic Instruction Weighting Module (DIWM). It ensures dynamic alignment between the global instructions and the agent's local observations. By applying a dynamic attention mechanism, DIWM adjusts the weighting of instruction words based on the agent's current visual observations, including its surroundings, objects, and possible actions. This process ensures that the most relevant words of the instruction are emphasized, helping the agent focus on the aspects of the instruction that directly correlate with its immediate visual perception. Through continuous updates to the instructions in response to environmental changes, DIWM facilitates a more effective alignment of the agent's visual and linguistic inputs. Our primary contributions can be summarized as follows:

- To the best of our knowledge, we are the first to address the misalignment between textual and visual information in long-instruction VLN, significantly enhancing the agent's abilities in path planning and decision-making.
- We design the Action-Perception Alignment Framework (APAF), which consists of two key modules, ACEM and DIWM. The former enriches the visual representation by incorporating action information that is often overlooked in existing methods. The latter adjusts the importance of instruction words at each time step, ensuring better alignment between the agent's visual observations and the instructions.
- Through extensive experiments, we demonstrate the effectiveness of our method and achieve state-of-the-art results, with improvements of 8.5% and 4.0% in Success Rate on the long-instruction R4R [32] and RxR [36] datasets, respectively.

The paper is organized as follows: Section 2 reviews the most relevant prior research. In Section 3, we introduce the Action-Perception Alignment Framework. Our experimental results are presented in Section 4, and Section 5 concludes the paper with a discussion on future research directions.

2 RELATED WORK

2.1 Vision-and-Language Navigation

With technological advancements, VLN has garnered increasing attention in recent years [12, 26, 55, 61, 65, 67, 71]. Early works [21, 74] in VLN predominantly employed RNNs to store historical actions and observations into the hidden states, facilitating accurate decision-making. To effectively capture the layouts of environments, Wang *et al.* [65] utilize a structured scene memory that precisely stores sensory information during navigation. Tan *et al.* [63] introduce a dual-phase training method aimed at boosting the agent's ability to generalize. Ma *et al.* [47] employ a progress monitor as an adaptive heuristic to aid in navigation search. With the advent of transformer-based architectures [17, 34], there has been a significant shift in how information is preserved and processed in VLN tasks. PRESS [39] introduces a stochastic sampling method aimed at narrowing the substantial disparity between expert-driven actions during training and actions sampled during testing. VLN-BERT [27] integrates recurrent units within a transformer framework to facilitate action prediction. Zhao *et al.* [72] proposes to explicitly predict the target location along a reference path to reduce the gap between agent performance and the oracle success rate in VLN.

Moreover, recent studies have also explored the use of topological graph [3, 9, 13, 16, 40, 51] and semantic maps [2, 4, 8, 10, 23, 28, 31, 45, 64, 70] to explicitly preserve various environmental information. ETPNav [3] conducts real-time topological mapping of environments by autonomously organizing predicted waypoints on a

traveled route. BEVBert [2] constructs a local semantic map to systematically consolidate partial observations and eliminate redundancies, concurrently representing navigational dependencies within a comprehensive global topological map. ACK [48] enhances both visual representation and action reasoning by incorporating commonsense knowledge as a spatio-temporal graph, thereby improving instruction grounding in complex environments. MiC[57] leverages large language models through an interactive prompting framework to dynamically generate scene-aware navigation plans under high-level instructions in the REVERIE task. Additionally, data augmentation techniques have been increasingly adopted to address the issues of dataset sparsity that often lead to model overfitting [12, 41, 69]. AutoVLN [12] generates a substantial VLN dataset using 900 unlabeled 3D buildings sourced from HM3D. ScaleVLN [69] further introduces a robust framework for producing extensive datasets to train agents. However, these methods have not been optimized for the long-instruction VLN task, and performance in such contexts is inadequate.

2.2 Visual Representations

In vision-and-language navigation, accurate decision-making heavily relies on the quality of visual representations at each step. Most studies [40, 66] encode panoramic images viewed by the agent at its current location into visual features, aligning these with room-related words in the instructions. Additionally, some research [13–15] utilizes object detectors to obtain information about objects at these locations, enriching the visual data. Similarly, both RelGraph [26] and OAAM [53] enhance visual representations by capturing the agent's current heading and elevation angles. Moreover, recent advancements have included constructing 2D occupancy grids or 3D point cloud maps to extract environmental information [19, 20, 44], addressing visual feature deficiencies. ORIST [52] incorporates room and object information and predicts the relative direction (e.g., left/right/front/back) for each candidate based on current visual features, aiming to align with direction cues in instructions. Different them, our method explicitly encodes the fine-grained heading and elevation angles between the agent and each candidate viewpoint to extract precise action features. This richer, action-aware visual representation ensures more effective alignment with action-oriented language instructions.

2.3 Textual Representations

Early studies [24, 27] in VLN treated instructions as static, remaining unchanged throughout the navigation process. Agents were reliant on temporal models, such as LSTM or transformers, to implicitly learn which words in the instructions were relevant to their current location [11, 41]. Some approaches have also proposed the construction of a Language Attention Graph, where nodes represent specific entities in the instructions (e.g., rooms, objects, actions) and edges depict relationships between these entities [26]. ROAA [22] utilize a transformer-encoder to develop an instruction extraction module that discriminates object and room meanings. DNA [7] uses a direction encoder to extract directional words from the instructions. ORIST [52] encodes instructions in a unified representation that remains fixed during navigation. In contrast, our work leverages both the agent's current observations and future action cues to dynamically adjust the weight of instruction words at each time step.

3 METHOD

In long-instruction VLN, the navigation environment is represented as an undirected graph $\mathcal{G}$, where nodes represent specific viewpoints, and edges represent feasible paths between them. At the start of navigation, the agent is initialized at a certain node in the graph, and a long instruction $\mathcal{W} = \{\boldsymbol{w}_i\}_{i=1}^{L}$, composed of L words, is provided. At each time step t, the agent receives a panoramic view $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{36}\}$ of its current location, which consists of 36 sub-images covering different viewing angles. Based on both $\mathcal{W}$ and $\mathcal{V}$, the agent is tasked

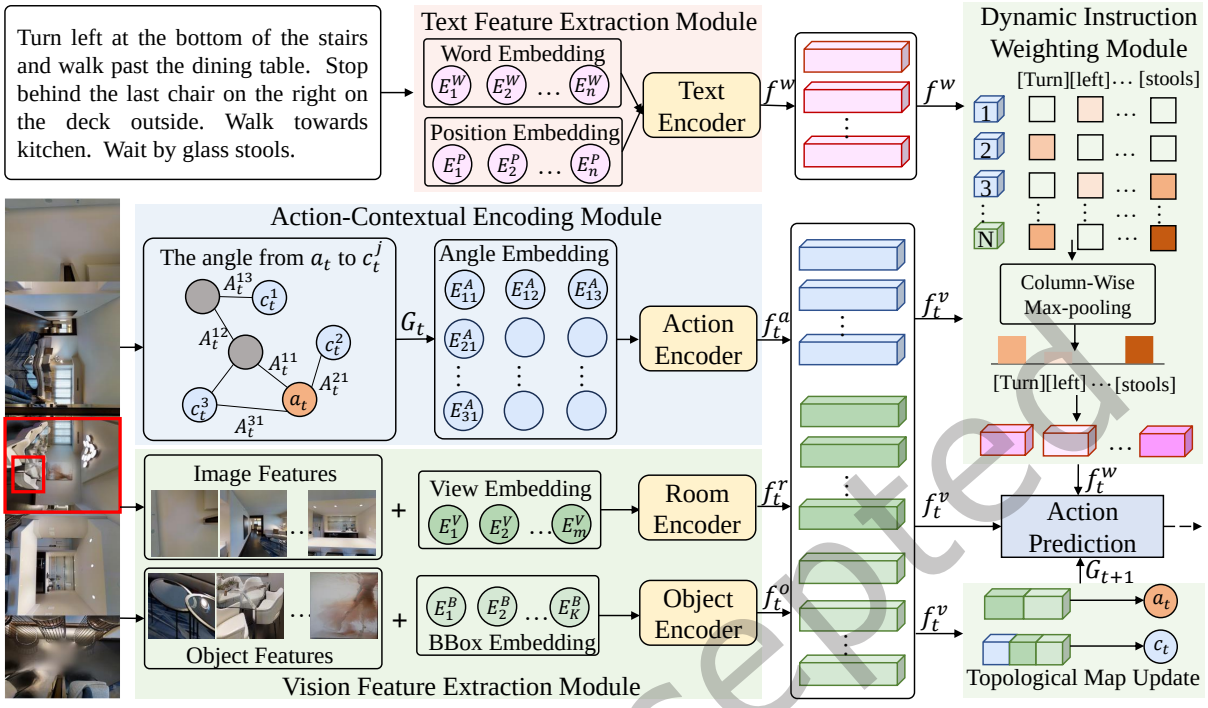


Fig. 2. The overall network architecture. At each time step t, our approach firstly employs two encoders to capture features about the current rooms and objects based on visual observations. In the Action-Contextual Encoding Module, we construct a topological map using the agent's exploration history. We calculate the angles required for the agent to turn toward each candidate location based on its current heading, elevation, and the spatial relationship with these points. Then these angles are encoded as features, and an action encoder extracts the corresponding action information. During the topological map update, the agent's current position is updated with room and object features, while candidate points are updated using room, object, and action features. In the Dynamic Instruction Weighting Module, we adjust the importance of instruction words based on room, object, and action information to maintain alignment with the agent's observations.

with selecting one of the available candidate locations as its next move or deciding to stop. The navigation is considered successful if the agent terminates at a position within 3 meters of the ground-truth viewpoint.

3.1 Overview of Our Method

The primary goal of this paper is to ensure the accurate alignment of textual and visual features in the long-instruction VLN task. As shown in Fig. 2, since navigation instructions typically include words related to rooms, objects, and actions, we also extract these three types of features from the agent's visual observations at each time step t. First, we design a Visual Features Extraction Module to extract room and object features at time t, denoted as $\boldsymbol{f}_t^r$ and $\boldsymbol{f}_t^o$, respectively. Then to obtain the action features $\boldsymbol{f}_t^a$, we construct a topological graph $\mathcal{G}_t \in \mathcal{G}$ based on the agent's trajectory up to time step t. In this graph, gray nodes represent previously visited locations, the brown node $\boldsymbol{a}_t$ indicates the agent's current location, and the blue nodes $\boldsymbol{c}_t = \{c_t^1, c_t^2, \ldots, c_t^k\}$ represent the candidate next locations. By calculating the angles the agent must rotate from $\boldsymbol{a}_t$ to each candidate location in $\boldsymbol{c}_t$, we encode the corresponding action features, denoted as $\boldsymbol{f}_t^a$. The three features $\boldsymbol{f}_t^r$, $\boldsymbol{f}_t^o$ and $\boldsymbol{f}_t^a$ are then

concatenated to form the current visual feature $\boldsymbol{f}_t^v$, which is used to update the corresponding nodes in $\mathcal{G}_t$, resulting in the updated topological map $\mathcal{G}_{t+1}$.

In addition, the instruction $\mathcal{W}$ corresponds to the entire navigation trajectory, while $\boldsymbol{f}_t^v$ only represents visual information at time t, creating a potential gap. To this end, we first encode $\mathcal{W}$ into a feature $\boldsymbol{f}^w$ and then utilize the Dynamic Instruction Weighting Module to generate a new instruction feature $\boldsymbol{f}_t^w$ by weighting each token in $\boldsymbol{f}^w$ using an attention mechanism. Finally, the visual feature $\boldsymbol{f}_t^v$, the updated instruction feature $\boldsymbol{f}_t^w$, and the topological graph $\mathcal{G}_{t+1}$ are fed into a decision network to predict the next action.

3.2 Vision Feature Extraction

At time step t, the agent receives a panoramic image of its current location, consisting of 36 views. We first utilize the image encoder from CLIP/B16 [59] to extract features for each view, as formulated below:

$$\boldsymbol{f}_r^i = \text{CLIP}\left(\boldsymbol{v}_i\right), \tag{1}$$

where $\boldsymbol{v}_i$ denotes the image from the i-th view. Next, we compute the view embedding for each image and add it to the corresponding feature $\boldsymbol{f}_r^i$. The resulting features are then passed into a room encoder. Specifically, the view embedding is represented as:

$$\boldsymbol{e}_v^i = \text{Embedding}\left(i\right), \tag{2}$$

and the room feature $\boldsymbol{f}_t^r$ is calculated as:

$$\boldsymbol{f}_t^r = \left\{g_r\left(\boldsymbol{f}_r^i + \boldsymbol{e}_v^i\right)\right\}_{i=1}^{36}, \tag{3}$$

where $\boldsymbol{f}_t^r \in \mathbb{R}^{L_r \times D}$, and L_r and D denote the number of views and feature dimension, respectively. Here, g_r represents the room encoder, which is implemented using a transformer-based architecture.

Similarly, to obtain object features, we first apply an object detection network [60] to the panoramic image $\mathcal{V}$ to generate bounding boxes $\{\boldsymbol{b}_i\}_{i=1}^m$ for all objects, where m represents the total number of objects in the panoramic image. Based on these bounding boxes, we crop the corresponding object images $\{\boldsymbol{o}_i\}_{i=1}^m$. Following the same approach as in eq. (1), we extract feature $\boldsymbol{f}_o^i$ for each object, denoted as:

$$\boldsymbol{f}_o^i = \text{CLIP}\left(\boldsymbol{o}_i\right). \tag{4}$$

Next, we compute the positional embedding $\boldsymbol{e}_b^i$ for each object's bounding box using a fully connected layer, formulated as:

$$\boldsymbol{e}_b^i = \text{FC}\left(b_i^0, b_i^1, b_i^2, b_i^3\right), \tag{5}$$

where $\boldsymbol{b}_i = \left(b_i^0, b_i^1, b_i^2, b_i^3\right)$ represents the coordinates of the top-left and bottom-right corners of the i-th bounding box. Finally, we combine $\boldsymbol{f}_o^i$ and $\boldsymbol{e}_o^i$, and pass them into the object encoder to obtain the object features $\boldsymbol{f}_t^o$:

$$\boldsymbol{f}_t^o = \left\{g_o\left(\boldsymbol{f}_o^i + \boldsymbol{e}_b^i\right)\right\}_{i=1}^m, \tag{6}$$

where $\boldsymbol{f}_t^o \in \mathbb{R}^{L_o \times D}$, with L_o denoting the number of detected objects. Here, g_o represents the object encoder, which shares the same transformer-based architecture as the room encoder g_r.

3.3 Action-Contextual Encoding

The features obtained in Section 3.2 are primarily aligned with room- and object-related words in the instructions, leaving many action-related words without corresponding visual representations. To address this issue and enrich the visual features with corresponding action information, we calculate the angles that the agent must rotate from its current position to each candidate location in the topological map $\mathcal{G}_t$.

As shown in Fig. 3, for the path from the current position n_1 to any candidate position n_4, we need to calculate the angles the agent must rotate when moving between two consecutive points along the path. Suppose the agent moves from node n_1 to node n_2, with the coordinates of these two points being (x_1, y_1, z_1) and (x_2, y_2, z_2),

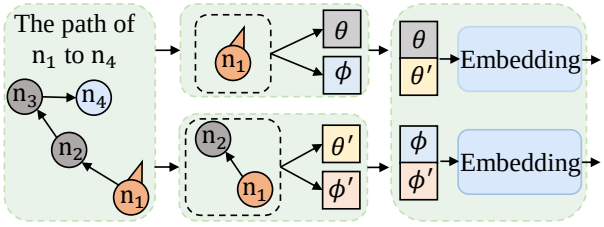


Fig. 3. For the path from the agent's current position n_1 to a candidate position n_4, we compute the relative heading θ' and elevation ϕ' between adjacent nodes along the path. These relative angles are then combined with the agent's current heading θ and elevation ϕ to obtain the corresponding embeddings.

respectively. The current heading and elevation angles of the agent are denoted as θ and ϕ respectively. To compute the required relative heading angle $\Delta\theta$, the formula is given as:

$$\theta' = p\left(\arctan2\left(x_2 - x_1, y_2 - y_1\right)\right), \tag{7}$$

$$\Delta\theta = \left(\left(\theta' - \theta\right) + \theta_{max}\right) \bmod \theta_{max}, \tag{8}$$

where $p\left(\cdot\right)$ converts the angle from radians to degrees, θ' represents the heading angle between the two nodes, and θ_{max} is a positive integer. Similarly, to obtain the relative elevation angle $\Delta\phi$, the formula is:

$$\phi' = p\left(\arctan2\left(z_2 - z_1, \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}\right)\right), \tag{9}$$

$$\Delta\phi = \max\left(-\phi_{max}, \min\left(\phi' - \phi, \phi_{max}\right)\right), \tag{10}$$

where ϕ' is the elevation angle of the adjacent nodes, and ϕ_{max} indicates the max value of relative elevation.

After getting the relative heading and elevation angles, we update θ and ϕ to θ' and ϕ', respectively. Subsequently, we continue calculating the relative heading and elevation angles for the remaining consecutive points along the path as shown in eq. (7) to eq. (10). This results in two sets of angles, $\Delta\Theta = \{\Delta\theta_1, \Delta\theta_2, \cdots, \Delta\theta_r\}$ and $\Delta\Phi = \{\Delta\phi_1, \Delta\phi_2, \cdots, \Delta\phi_r\}$, representing the relative heading and elevation angles for the entire path, where r denotes the number of edges along the path.

We then derive the action features for the path by embedding these angle sequences, formulated as:

$$\boldsymbol{e}_a^i = \text{Embedding}\left(\Delta\Theta\right) + \text{Embedding}\left(\Delta\Phi\right), \tag{11}$$

where e_a^i represents the embedding features for the i-th path. These features are then passed through an action encoder to generate the final action features:

$$\boldsymbol{f}_t^{a_i} = \text{AvgPooling}\left(g_a\left(\boldsymbol{e}_a^i\right)\right), \tag{12}$$

$$\boldsymbol{f}_t^a = \left[\boldsymbol{f}_t^{a_1}, \boldsymbol{f}_t^{a_2}, \cdots, \boldsymbol{f}_t^{a_n}\right], \tag{13}$$

where $\boldsymbol{f}_t^a \in \mathbb{R}^{L_n \times D}$, with L_n denoting the number of candidate positions the agent can reach. Here, g_a represents an action encoder.

After obtaining all three types of features in Section 3.2 and Section 3.3, we concatenate them to form the feature $\boldsymbol{f}_t^o = \left[\boldsymbol{f}_t^r, \boldsymbol{f}_t^o, \boldsymbol{f}_t^a\right] \in \mathbb{R}^{(L_r + L_o + L_n) \times D}$. Then it is used to update $\mathcal{G}_t$, resulting in a new topological map $\mathcal{G}_{t+1}$. Specifically, for the current position of the agent, we use $\boldsymbol{f}_t^r$ and $\boldsymbol{f}_t^o$ as its features. For the candidate point c_t^i, we use the room feature $\boldsymbol{f}_t^{r\prime} \in \boldsymbol{f}_t^r$ and the object feature $\boldsymbol{f}_t^{o\prime} \in f_t^o$, both corresponding to those visible view, along with the action feature $\boldsymbol{f}_t^{a_i}$ as its features.

3.4 Dynamic Instruction Weighting

Since $\boldsymbol{f}_t^v$ represents information at only a specific time step t, and the instruction $\mathcal{W}$ describes the entire navigation trajectory, $\boldsymbol{f}_t^v$ corresponds to a subset of words in $\mathcal{W}$. In this paper, based on the previously obtained visual and action features, we apply an attention mechanism to compute the importance of each word at the current time step. By dynamically updating the instruction, this approach helps reduce the gap and improve navigation accuracy.

First, we obtain the embedding $\boldsymbol{e}_w^i$ for each word in the instruction and its corresponding positional encoding $\boldsymbol{e}_p^i$. These embeddings are then passed through the text encoder to obtain the instruction feature $\boldsymbol{f}^w \in \mathbb{R}^{L\times D}$, as shown in the following formula:

$$\boldsymbol{f}^w = \text{Text-Encoder}\left(\boldsymbol{e}_w^i + \boldsymbol{e}_p^i\right). \tag{14}$$

Next, we apply an attention mechanism based on the visual feature $\boldsymbol{f}_t^v$ to calculate the weight A_t of each word at the current time step:

$$A_t = \text{softmax}\left(\frac{\boldsymbol{f}_t^v\,(\boldsymbol{f}^w)^T}{\sqrt{D}}\right). \tag{15}$$

We then perform max pooling on A_t to obtain the final attention weights A_t':

$$A_t' = \text{MaxPooling}(A_t). \tag{16}$$

Finally, the instruction feature at the current time step t, denoted as $\boldsymbol{f}_t^w \in \mathbb{R}^{L\times D}$, is computed as:

$$\boldsymbol{f}_t^w = \left\{A_t'(i)\cdot \boldsymbol{f}^w(i)\right\}_{i=1}^{L}. \tag{17}$$

3.5 Action Prediction

We follow DUET [13] for action prediction. In the graph $\mathcal{G}_{t+1}$, each node stores the room type, object features, and action-related information observed at that location. During decision-making, the agent uses these node features to predict the probability of moving to each candidate node. Specifically, for all candidate nodes $c_{t+1} \in \mathcal{G}_{t+1}$, we use a *Coarse-scale Cross-modal Encoder* to predict the probability that the agent reaches each node. This prediction is based on each candidate node's stored features and the current instruction feature $\boldsymbol{f}_t^w$, resulting in a probability set $\boldsymbol{s} = \{s_0, s_1, s_2, \ldots, s_{k_1}\}$, where k_1 denotes the number of candidate nodes. For the agent's adjacent nodes at time t, we apply a *Fine-scale Cross-modal Encoder*, which takes as input the instruction feature $\boldsymbol{f}_t^w$ and the visual feature $\boldsymbol{f}_t^v$. It outputs a probability set $\boldsymbol{q} = \{q_0, q_1, q_2, \ldots, q_{k_2}\}$ representing the likelihood of reaching each adjacent node. Finally, the coarse-scale and fine-scale probabilities $\boldsymbol{s}$ and $\boldsymbol{q}$ are fused to produce the final navigation distribution. The agent selects the node with the highest probability as the next step.

3.6 Training and Inference

3.6.1 Pretraining. Following prior work in the field [25, 35, 51], the benefits of pretraining transformer-based models for VLN tasks are well-documented. In our approach, we adopt two auxiliary tasks to pretrain the proposed model.

Masked Language Modeling (MLM). MLM is a widely adopted auxiliary task for pre-training BERT-like models [17]. In the context of long-instruction VLN, MLM is used to predict masked words $\mathcal{W}_m$ by leveraging contextual information from the unmasked words $\mathcal{W}_{\setminus m}$, along with the visual features $\boldsymbol{f}_t^v$, instruction features $\boldsymbol{f}_t^w$, and the topological map $\mathcal{G}_{t+1}$. Specifically, we randomly mask 15% of the tokens in the input instruction and extract features as described in previous sections. These extracted features are then passed into the MLM head,

which consists of two fully connected layers. The objective of the task is to minimize the negative log-likelihood as follows:

$$L_{\text{MLM}} = -\mathbb{E}_{(\mathcal{W},)\sim\mathcal{D}} \log \mathcal{P}_\eta \left(\mathcal{W}_m | \mathcal{W}_{\setminus m}, \boldsymbol{f}_t^v, \boldsymbol{f}_t^w, \mathcal{G}_{t+1}\right) \tag{18}$$

where $\mathcal{D}$ represents the training dataset, refers to the agent's trajectory, and η is the learnable parameters.

Single-step Action Prediction (SAP). In the SAP task [11], the agent is tasked with predicting its next action based on the preceding steps. The task is optimized via a cross-entropy loss:

$$L_{\text{SAP}} = -\mathbb{E}_{(\mathcal{W},,a_t^*)\sim\mathcal{D}} \log \mathcal{P}_\eta \left(a_t^* | \mathcal{W}, \boldsymbol{f}_t^v, \boldsymbol{f}_t^w, \mathcal{G}_{t+1}\right) \tag{19}$$

where a_t^* is the teacher action at the next step.

3.6.2 Fine-tuning and Inference. During the trajectory rollout phase of fine-tuning, we alternate between two stages: teacher-forcing and student-forcing. In the teacher-forcing approach, as illustrated in eq. (19), the agent consistently executes actions provided by the teacher. In contrast, during the student-forcing phase, where the ground-truth path is unavailable, a pseudo label is generated to provide supervision. The pseudo label a_t^{p*} is obtained using a pseudo-interactive demonstrator (PID) [13], which selects the pseudo action based on two possible strategies, referred to as Strategy A and Strategy B, respectively. The former is to choose the candidate node that minimizes the overall distance from the current position to the final destination. The latter involves selecting the candidate node that minimizes the normalized Dynamic Time Warping (nDTW) [30] distance between the agent's traversed path and the target path.

At each decision step, the agent samples the next action from the predicted score distribution, with supervision provided by either the teacher's action a_t^* or the pseudo label a_t^{p*}. The loss function for student-forcing is defined as follows:

$$L_{\text{PID}} = -\mathbb{E}_{(\mathcal{W},,a_t^{p*})\sim\mathcal{D}} \log \mathcal{P}_\eta \left(a_t^{p*} \mid \mathcal{W}, \boldsymbol{f}_t^v, \boldsymbol{f}_t^w, \mathcal{G}_{t+1}\right) \tag{20}$$

To train the agent effectively, we integrate supervision from both the teacher-forcing and student-forcing stages. The final combined loss function is:

$$L = \lambda L_{\text{SAP}} + L_{\text{PID}} \tag{21}$$

where λ is a weighting factor that balances the contributions of the two loss components.

For inference, the agent predicts an action at each step and moves accordingly unless it predicts a stop action. If the number of steps exceeds a predefined limit, the agent is forced to stop, and the location with the highest stop probability is taken as the final position.

4 EXPERIMENTS

4.1 Datasets

In this paper, we focus primarily on the long-instruction vision-and-language navigation task. Consequently, we evaluate our model on two existing long-instruction datasets: R4R [32] and RxR [36]. Additionally, we also conduct experiments on a commonly used fine-grained dataset, R2R [5], to assess the generalization capability of our model.

R2R is an indoor dataset constructed using the Matterport3D [6] simulator, featuring 90 photo-realistic houses and 10,567 panoramic images. The dataset provides 7,189 navigation trajectories, each accompanied by three corresponding natural language instructions. In total, it contains 21,567 unique words. Both the instructions and trajectories are relatively short, with an average instruction length of 29 words. The average path length is 10 meters, and the agent typically needs to take 4 to 6 steps to complete the trajectory.

R4R is an extended version of the R2R dataset, designed to address long-instruction navigation task. It is created by concatenating two adjacent R2R trajectories in a tail-to-head manner, resulting in longer paths. Consequently,

R4R features both longer instructions and more complex trajectories. Additionally, the trajectories are less biased, as they are not necessarily the shortest path between the start and end points.

RxR highlights the importance of language in VLN by addressing path biases and providing richer descriptions of visible entities compared to R2R. It includes longer navigation paths, with each path spanning multiple rooms, requiring the agent to maintain long-term context tracking throughout the task. In this paper, we utilize the all English instructions from the RxR dataset to evaluate our model.

4.2 Evaluation Metrics

To evaluate the effectiveness of our approach against prior methods, we employ the standard evaluation metrics commonly used in visual-and-language navigation tasks, as described in [5, 54]. These metrics include: (1) Trajectory Length (TL), which measures the agent's average path length in meters; (2) Navigation Error (NE), representing the average distance in meters between the agent's final position and the target; (3) Success Rate (SR), defined as the percentage of instructions completed successfully, where NE is less than 3 meters; (4) Oracle SR (OSR), the SR calculated assuming an optimal stop policy; (5) Success weighted by Path Length (SPL), which adjusts SR according to the efficiency of the path; (6) Coverage weighted by Length Score (CLS), which evaluates the ratio of the agent's traversed path that overlaps with the optimal path; (7) Normalized Dynamic Time Warping (nDTW), which measures the alignment between the agent's trajectory and the ground-truth path and (8) Success rate weighted normalized Dynamic Time Warping (SDTW), which adjusts nDTW by factoring in the SR. For all metrics except TL and NE, higher values indicate better performance.

4.3 Implementation Details

4.3.1 Model Architectures. The Faster-RCNN model used for extracting object features was pre-trained on the COCO [42] dataset. The room, object, and action encoders all consist of two layers of transformer networks. The text encoder follows the configuration from DUET [13], comprising a 9-layer transformer structure. The method for Action Prediction is also consistent with DUET [13]. For the R2R and R4R datasets, the model is initialized using a pre-trained LXMERT [62] model, while for the RxR dataset, a pre-trained RoBERTa [46] model is used for initialization.

4.3.2 Training Details. For pre-training, we set the batch size to 32 and use 4 NVIDIA RTX 3090 GPUs, with a learning rate of $5e^{-4}$ and a total of 200k iterations. During fine-tuning, the batch size is reduced to 4, the learning rate is set to $1e^{-5}$, and the number of iterations is 200k. The instruction feature $\boldsymbol{f}^w$ is updated at every time step. The update is based on current visual observations, including room types, objects, and action-related information. We adopt the Adam optimizer without weight decay and without learning rate warm-up. Gradient clipping is applied with a maximum norm of 40. The dropout rate in the transformer layers is set to 0.5. The dimension D of both the feature $\boldsymbol{f}_t^w$ and $\boldsymbol{f}_t^v$ is denoted as 768. During navigation, the maximum number γ of steps the agent can take is set to 20. The maximum values of the angles θ_{max} and ϕ_{max} are 360 and 30, respectively. For the R4R dataset, the agent's next candidate node is chosen from any node in the topological map, whereas for the R2R and RxR datasets, only unvisited nodes are considered as candidate nodes. In the R2R dataset, we generate pseudo labels using Strategy A, with λ set to 0.15. For the R4R and RxR datasets, Strategy B is employed, and λ is set to 0.8.

4.4 Performance Comparison

We present the comparisons with previous state-of-the-art approaches on the R4R, RxR, and R2R datasets. The primary metrics for each dataset are emphasized in bold.

As shown in Table 1 and Table 2, we conducted experiments on two long-instruction datasets, R4R and RxR. The results indicate that our model demonstrates significant improvements across all metrics. On the R4R dataset,

Table 1. Comparison with state-of-the-art methods on the R4R dataset. † indicates results reproduced under the same parameter settings as the original method. For our method, we conduct 5 repeated runs using the runtime timestamp as the random seed, and report the mean and standard deviation of the results.

Methods	Validation Seen						Validation Unseen					
	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
Speaker-Follower [21]	5.35	51.9	37.3	46.4	-	-	8.47	23.8	12.2	29.6	-	-
RCM-a (goal) [74]	5.11	55.5	32.3	40.4	-	-	8.45	28.6	10.2	20.4	-	-
RCM-a (fidelity) [74]	5.37	52.6	30.6	55.3	-	-	8.08	26.1	7.7	34.6	-	-
RCM-b (goal) [30]	-	-	-	-	-	-	-	28.7	15.0	33.4	26.9	11.4
RCM-b (fidelity) [30]	-	-	-	-	-	-	-	28.5	21.4	35.4	30.4	12.6
PTA (high-level) [37]	4.54	58	39	60	58	41	8.25	24	10	37	32	10
EGP [16]	-	-	-	-	-	-	8.00	30.2	-	44.4	37.4	17.5
SSM [65]	4.60	63	-	65	56	44	8.27	32	-	53	39	19
RelGraph [26]	5.31	52	46	55	62	50	7.43	36	26	41	47	34
RecBERT [27]	-	-	-	-	-	-	6.67	43.6	-	51.4	45.1	29.9
HAMT [11]	-	-	-	-	-	-	6.09	44.6	-	57.7	50.3	31.8
HAMT †	5.44	53.7	50.7	67.5	61.9	44.9	6.69	40.7	38.3	58.9	50.6	30.3
LANA [68]	-	-	-	-	-	-	-	43.2	-	59.7	52.3	31.7
ScaleVLN [69]	-	-	-	-	-	-	6.09	44.2	-	59.6	52.8	32.7
ScaleVLN †	5.36	53.6	48.8	64.0	57.7	41.9	5.90	49.1	43.4	59.0	53.2	35.8
BSG [45]	-	-	-	-	-	-	6.12	47	-	59	53	34
VER [44]	-	-	-	-	-	-	6.10	47	-	61	54	33
Ours	**3.29** ±0.099	**69.2** ±1.31	**65.0** ±1.00	**69.3** ±1.23	**66.6** ±0.89	**53.9** ±1.28	**4.97** ±0.088	**55.5** ±1.21	**51.1** ±1.45	**62.3** ±0.43	**57.6** ±1.06	**40.3** ±0.90

Table 2. Comparison with state-of-the-art methods on the RxR dataset using English instructions. For our method, we conduct 5 repeated runs using the runtime timestamp as the random seed, and report the mean and standard deviation of the results.

Methods	Validation Seen					Validation Unseen				
	SR↑	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
EnvDrop [63]	48.1	44	61	57	40	38.5	34	54	51	32
Syntax [38]	48.1	44	61	58	40	39.2	35	56	52	32
SOAT [50]	-	-	-	-	-	44.2	-	54.8	36.4	
HOP [55]	49.4	45.3	-	58	40	42.3	36.3	-	52	33
HOP+ [56]	53.6	47.9	-	59	43	45.7	38.4	-	52	36
FOAM [18]	-	-	-	-	-	42.8	38.7	-	54.1	35.6
ADAPT (ResNet152) [41]	52.7	47.0	61.3	58.5	42.9	46.7	40.3	56.6	53.6	37.3
ADAPT (CLIP) [41]	50.3	44.6	59.6	56.3	40.6	46.9	40.2	57.2	54.1	37.7
GOAT [66]	74.1	68.1	-	71.0	61.4	68.2	61.7	-	67.1	56.6
Ours	**76.5** ±0.75	**71.2** ±0.24	**75.2** ±0.56	**75.4** ±0.55	**67.1** ±0.27	**72.2** ±0.54	**65.0** ±0.24	**70.3** ±0.37	**70.3** ±0.22	**60.8** ±0.24

compared to the current state-of-the-art method, VER, we observe improvements of 8.5%, 3.6%, and 7.3% in the SR, nDTW, and SDTW metrics on the validation unseen split, respectively. These findings suggest that our approach enables the agent to more accurately follow instructions and efficiently adapt to the long-instruction vision-and-language navigation task. The results on the RxR dataset further corroborate this effectiveness.

Table 3. Comparison with state-of-the-art methods on the R2R dataset. For our method, we conduct 5 repeated runs using the runtime timestamp as the random seed, and report the mean and standard deviation of the results.

Methods	Validation Seen					Validation Unseen					Test Unseen				
	TL↓	NE↓	OSR↑	**SR↑**	**SPL↑**	TL↓	NE↓	OSR↑	**SR↑**	**SPL↑**	TL↓	NE↓	OSR↑	**SR↑**	**SPL↑**
Seq2Seq [5]	11.33	6.01	53	39	-	8.39	7.81	28	21	-	8.13	7.85	27	20	-
Speaker-Follower [21]	-	3.36	74	66	-	-	6.62	45	36	-	14.82	6.62	-	35	28
RCM [74]	10.65	3.53	75	67	-	11.46	6.09	50	43	-	11.97	6.12	50	43	38
SSM [65]	14.70	3.10	80	71	62	20.70	4.32	73	62	45	20.40	4.57	70	61	46
EnvDrop [63]	11.00	3.99	-	62	59	10.70	5.22	-	52	48	11.66	5.23	59	51	47
PREVALENT [25]	10.32	3.67	-	69	65	10.19	4.71	-	58	53	10.51	5.30	61	54	51
ORIST [52]	-	-	-	-	-	10.90	4.72	-	57	51	11.31	5.10	-	57	52
RelGraph [26]	10.13	3.47	-	67	65	9.99	4.73	-	57	53	10.29	4.75	61	55	52
NvEM [1]	11.09	3.44	-	69	65	11.83	4.27	-	60	55	12.98	4.37	66	58	54
AirBert [24]	11.09	2.68	-	75	70	11.78	4.10	-	62	56	12.41	4.13	-	62	57
VLNBERT [27]	11.13	2.90	-	72	68	12.01	3.93	-	63	57	12.35	4.09	70	63	57
MARVAL [33]	10.60	2.99	-	73	69	10.15	4.06	-	65	61	10.22	4.18	67	62	58
EnvMix [43]	10.88	2.48	-	75	72	12.44	3.89	-	64	58	13.11	3.87	72	65	59
HAMT [11]	11.15	2.51	-	76	72	11.46	2.29	-	66	61	12.27	3.93	72	65	60
SnapEnsemble [58]	-	-	-	-	-	12.05	3.63	-	67	60	12.71	3.82	-	65	60
HOP+ [56]	11.31	2.33	-	78	73	11.76	3.49	-	67	61	12.67	3.71	-	66	60
TD-STP [73]	-	2.34	83	77	73	-	3.22	76	70	63	-	3.73	72	67	61
DUET [13]	-	-	-	-	-	13.94	3.31	-	72	60	14.73	3.65	-	69	59
DNA [7]	-	3.20	77	69	65	-	4.93	61	52	44	-	5.15	60	53	46
KERM [40]	12.16	2.19	-	79.73	73.79	13.54	3.22	-	71.95	60.91	14.60	3.61	-	69.73	59.25
BEVBert [2]	13.56	2.17	88	81	74	14.55	2.81	84	75	64	15.87	3.13	81	73	62
GridMM [70]	-	-	-	-	-	13.27	2.83	-	75	64	14.43	3.35	-	73	62
ScaleVLN [69]	11.90	2.16	87	80	75	12.40	2.34	87	79	70	14.27	2.73	83	77	68
VER [44]	-	-	-	-	-	14.83	2.80	-	76	65	15.23	2.74	-	76	66
GOAT [66]	-	**1.79**	**88.64**	**83.74**	**79.48**	-	2.40	84.72	77.82	68.13	-	3.04	80.35	74.57	64.94
Ours	12.80 ±0.47	2.01 ±0.043	87.79 ±0.99	81.88 ±0.50	75.84 ±0.58	13.20 ±0.52	**2.27** ±0.059	**87.87** ±0.70	**80.19** ±0.54	**71.53** ±1.10	13.97 ±0.71	**2.47** ±0.052	**85.02** ±0.96	**78.32** ±0.14	**69.83** ±0.52

Additionally, we evaluated our model on a conventional fine-grained dataset, R2R. As shown in Table 3, we find that the improvements in key metrics, namely SR and SPL, were not as pronounced compared to existing methods. In relation to the ScaleVLN approach, our model only achieves increases of 1.32% and 1.83% in these two metrics on the test splits. We believe this limited enhancement is attributable to the relatively short instructions and trajectories present in the R2R dataset, which makes it easier for the agent to comprehend the instructions and align them with visual observations. Consequently, the features from both modalities have already aligned effectively in these methods, resulting in minimal improvement from the modules we designed.

4.5 Ablation Study

In this section, we demonstrate the contributions of each module within our model through experimental analysis. Unless otherwise specified, all experiments are conducted on the validation unseen split of R4R dataset.

4.5.1 Different types of features. We conducted experiments to analyze how different types of features affect agent navigation capabilities. As demonstrated in Table 4, we found that both object and action features significantly enhance the navigation ability of agents. Specifically, the inclusion of these features led to increases of 4.1% in SR and 3.2% in nDTW respectively, as shown in the last row of Table 4. Moreover, object and action features make distinct contributions to navigation improvements, detailed in the second and third rows of the table. The former primarily increases the success rate of navigation, enhancing the SR metric by 3.7%. Conversely, the latter predominantly improves the ability of agents to follow instructions, resulting in a 2.6% enhancement in the nDTW metric.

Table 4. A comparison examining how various feature types influence agent navigation. f_t^o and f_t^a represent the object and action features, respectively at time t. All experiments do not use the Dynamic Instruction Weighting Module.

f_t^o	f_t^a	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
×	×	5.66	50.1	45.9	60.5	52.0	35.2
✓	×	5.35	53.8	47.0	60.5	53.4	37.8
×	✓	5.47	52.5	46.4	**61.6**	54.6	38.2
✓	✓	**5.21**	**54.2**	**48.3**	61.1	**55.2**	**38.5**

Table 5. The influence of Dynamic Instruction Weighting Module on agent navigation. f_t^w indicates whether the updated instruction features are utilized to guide the agent. The term "top-k words" refers to the use of the k most weighted words from f_t^w to assist the agent in navigation.

f_t^w	Top-k words	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
×	-	5.21	54.2	48.3	61.1	55.2	38.5
✓	3	5.82	53.1	46.8	59.8	54.1	37.5
✓	5	5.34	52.5	47.3	58.5	54.7	37.3
✓	10	5.32	53.6	48.1	58.2	54.9	38.1
✓	20	5.16	**56.0**	48.6	60.8	55.5	39.5
✓	30	5.17	54.9	**49.5**	61.6	55.3	39.0
✓	all	**5.02**	55.5	48.7	**62.1**	**56.3**	**39.9**

4.5.2 Influence of Dynamic Instruction Weighting Module. Table 5 illustrates the impact of the Dynamic Instruction Weighting Module on navigation performance. Data from the first and last rows indicate that the use of this module leads to significant improvements in the three primary metrics SR, nDTW, and SDTW. Additionally, we explored the performance of the agent when only the k words with the highest weights are used as instructions. Results from Table 5 show that when k is set too low, the agent's success rate decreases. As k increases, the agent's performance gradually improves. This suggests that words that are not directly related to the current location are not entirely without effect. They can provide rich contextual information that aids in accurate navigation.

Table 6. The influence of view and bounding box embeddings.

view	bbox	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
×	×	5.35	54.6	48.1	**62.8**	55.1	38.2
✓	×	5.23	54.2	47.9	61.2	55.5	38.9
×	✓	**4.89**	55.3	48.6	61.4	55.9	39.4
✓	✓	5.02	**55.5**	**48.7**	62.1	**56.3**	**39.9**

4.5.3 Impact of different embeddings. In our encoding of room and object features, we incorporated view embeddings and bounding box embeddings, respectively. To analyze the effects of these embeddings, we conducted experiments as detailed in Table 6. The results show that after adding these two types of embeddings, there was a notable improvement in metrics such as SR and nDTW. We also found that, compared to view embeddings, bounding box embeddings had a greater impact on successful navigation. This could be due to object information appearing more frequently than room information.

Table 7. A comparison of various methods for obtaining action features.

Method	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
S	5.21	53.8	47.1	62.0	55.2	37.7
F	**5.02**	**55.5**	**48.7**	**62.1**	**56.3**	**39.9**

4.5.4 Different methods for obtaining action features. In our experiments, we evaluated two different methods for extracting action features, designated as methods *S* and *F*. The former directly computes the angles between the agent's current position and the candidate locations, while the latter calculates the angles between all adjacent points along the path to the candidate locations.

As demonstrated in Table 7, method *F* resulted in a more substantial improvement in navigation performance compared to method *S*. This is primarily because method *S* directly calculates the angle between the current position and candidate positions, which loses relevance when the distance between these points is excessive. In contrast, method *F* computes angles between adjacent points, effectively avoiding this limitation. Consequently, we adopted method *F* for our training regimen.

Table 8. The impact of heading and elevation angles on the agent's performance.

heading	elevation	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
×	×	5.38	53.9	47.2	60.1	53.6	37.9
×	✓	5.47	53.6	47.4	59.5	54.7	38.1
✓	×	5.07	54.8	48.1	61.8	56.1	38.6
✓	✓	**5.02**	**55.5**	**48.7**	**62.1**	**56.3**	**39.9**

4.5.5 The influence of heading and elevation angles for action features. Table 8 demonstrates the roles of heading and elevation angles in computing action features. We observed that action features derived solely from the elevation angle did not significantly enhance agent navigation. In contrast, action features obtained using the heading angle resulted in substantial improvements of 0.9% and 2.5% in SR and sDTW, respectively. We hypothesize that this is because instructions typically contain words related to left and right turns, which are associated with the heading angle, whereas words related to upward and downward movements, linked to the elevation angle, are less frequent. Finally, when both angles were used together to calculate action features, there was a further enhancement in navigation performance.

4.5.6 Different strategies of generating pseudo action. During the fine-tuning phase, we employed two methods to generate pseudo labels. Table 9 shows the performance of these two methods across three different datasets. We observed that for long-instruction datasets, such as R4R and RxR, using Strategy B for pseudo label generation yielded better results, particularly in terms of improvements in the nDTW and SDTW metrics. This indicates that this strategy helps the agent follow the instructions more effectively and reduces the confusion caused by lengthy instructions. On the other hand, for standard datasets like R2R, Strategy A performed better, enabling the agent to learn the shortest path from the starting point to the goal more effectively.

4.5.7 The impact of instruction processing methods on navigation. As shown in Table 10, we conducted an experiment to compare two different instruction processing methods. In method *M*, the long instruction is divided into multiple shorter instructions based on periods and provided sequentially. In method *N*, the agent is provided with the entire long instruction directly. We observed that method *M* resulted in a significant performance drop

Table 9. Different strategies of generating pseudo action. We conducted experiments on the validation unseen split of R4R, RxR and R2R datasets

Dataset	Strategy	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
R4R	A	5.36	53.1	**49.6**	54.0	50.2	36.4
	B	**5.02**	**55.5**	48.7	**62.1**	**56.3**	**39.9**
RxR	A	3.90	70.8	59.5	65.2	63.4	55.4
	B	**3.62**	**72.1**	**65.1**	**70.9**	**70.7**	**60.7**
R2R	A	**2.29**	**80**	**70**	**73.2**	**74.0**	**67.8**
	B	2.40	78	69	72.7	73.6	65.9

Table 10. The impact of instruction processing methods on navigation. We conducted experiments on the validation unseen split of R4R, RxR and R2R datasets

Dataset	Method	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
R4R	*M*	9.73	14.3	11.9	50.6	36.9	9.1
	N	**5.02**	**55.5**	**48.7**	**62.1**	**56.3**	**39.9**
RxR	*M*	9.60	34.1	24.6	38.9	31.0	21.5
	N	**3.62**	**72.1**	**65.1**	**70.9**	**70.7**	**60.7**
R2R	*M*	3.80	64.8	46.4	53.7	50.7	43.1
	N	**2.29**	**80**	**70**	**73.2**	**74.0**	**67.8**

compared to method *N*. This is because splitting the long instruction into shorter segments may cause the model to lose the contextual information required for global path planning. As a result, the agent may fail to recognize its location errors, leading to incorrect decisions. Without the global navigation context, the agent becomes overly reliant on local information, which increases the likelihood of abandoning the task when errors occur. This ultimately reduces the agent's ability to correct mistakes. Additionally, we also found that the decline in SR on the R2R dataset was much smaller compared to the two long-instruction datasets, R4R and RxR. This further suggests that the additional contextual information provided by longer instructions aids the agent in more accurately navigating.

4.6 Quantitative experiments

In this section, we conduct an experimental analysis of several hyperparameter settings used during the experiments, identifying the optimal configurations.

Table 11. The effect of the transformer block count in the room, object, and action encoders. The experiment was finished on the validation unseen split of R4R dataset.

Number of blocks	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
1	5.41	52.9	46.8	58.9	54.4	38.1
2	5.02	55.5	48.7	**62.1**	**56.3**	**39.9**
4	**4.94**	**55.9**	**49.1**	60.1	55.7	39.2

4.6.1 The number of blocks in three encoders. Table 11 presents the impact of using different numbers of transformer blocks for the room, object, and action encoders on the agent's navigation performance. As shown in the table, when only one transformer block is used, the agent's navigation success rate is suboptimal due to the limited number of parameters. However, when the number of blocks is increased to 2 or 4, navigation performance improves significantly. To reduce computational cost, we opted to use 2 transformer blocks during training.

Table 12. The impact of the value of γ on agent navigation. We conduct experiments on the validation unseen split of R4R, RxR, and R2R datasets

Dataset	γ	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
R4R	15	5.32	53.1	48.2	57.3	52.6	36.8
	20	**5.02**	55.5	**48.7**	62.1	**56.3**	39.9
	25	5.06	**56.7**	48.3	**62.6**	56.0	**40.2**
RxR	15	**3.54**	71.8	62.9	68.2	67.1	58.1
	20	3.62	**72.1**	**65.1**	70.9	**70.7**	**60.7**
	25	3.75	71.6	64.8	**71.3**	69.7	59.6
R2R	15	2.40	79	70	71.5	**74.0**	67.0
	20	**2.29**	**80**	**70**	73.2	**74.0**	**67.8**
	25	2.43	**80**	68	**73.9**	71.9	65.1

4.6.2 The maximum number of steps the agent can take. The maximum number of steps γ that the agent can take during navigation also has a significant impact on its performance. Table 12 shows the results for different values of γ. We observed that moderately increasing γ, for example, from 15 to 20, leads to a noticeable improvement in navigation success rates. This is because a larger γ allows the agent to explore a wider area, increasing the likelihood of selecting the correct destination. However, further increasing γ does not result in additional improvements and may even degrade performance. We attribute this to the fact that, with an excessively large γ, the agent's path becomes longer and deviates more from the ground-truth path, leading to a decline in some performance metrics.

4.6.3 The weight of the teacher-forcing stage. As shown in eq. (21), we train the agent in the teacher-forcing and student-forcing stages. λ is the balancing factor, where a larger λ value indicates a greater emphasis on the teacher-forcing stage. Table 13 presents the agent's performance on different datasets as λ varies. We found that the impact of λ on navigation performance is significant, particularly for the nDTW and SDTW metrics. For long-instruction datasets, a larger λ produces the best results. In contrast, for standard datasets, a smaller λ performs better. This is because, in long-instruction datasets, both the trajectories and instructions are longer and more complex. A larger λ ensures that the agent relies more heavily on the correct path during training. This reduces error accumulation over long sequences, enabling the agent to follow complex instructions more accurately. For standard datasets, where the navigation paths and instructions are more straightforward, a smaller λ allows the model to rely more on its own predictions. This better prepares it for real-world scenarios where autonomous decision-making is required. Since the paths are shorter, error accumulation is less of an issue, and the agent can recover from mistakes more easily.

Table 13. The influence of the value of λ on agent navigation. All results were obtained from validation unseen split R4R, RxR, and R2R datasets.

Dataset	λ	NE↓	**SR↑**	SPL↑	CLS↑	**nDTW↑**	**SDTW↑**
R4R	0.15	5.15	52.9	46.4	52.0	43.3	30.4
	0.5	5.08	54.7	47.3	57.5	53.6	38.5
	0.8	**5.02**	**55.5**	**48.7**	**62.1**	**56.3**	**39.9**
RxR	0.15	3.90	70.8	59.5	61.3	63.4	55.4
	0.5	4.05	69.4	62.1	67.1	67.2	57.4
	0.8	**3.62**	**72.1**	**65.1**	**70.9**	**70.7**	**60.7**
R2R	0.15	**2.29**	**80**	**70**	73.2	**74.0**	**67.8**
	0.5	2.36	**80**	69	**73.9**	73.3	66.6
	0.8	2.61	78	68	73.0	71.7	64.5

Walk down the stairs, and turn left to the room under the stairs and stop in the doorway. Leave the office and take a right. Go past the stairs, and veer right. Stop next to the statue of the creature on the bike.

Walk down the stairs, and turn left to the room under the stairs and stop in the doorway. Leave the office and take a right. Go past the stairs, and veer right. Stop next to the statue of the creature on the bike.

…, wait by the counter. Walk past the dual ovens and refrigerator and continue forward. Wait in front of the bedroom door.

…, wait by the counter. Walk past the dual ovens and refrigerator and continue forward. Wait in front of the bedroom door.

Walk out of the bathroom door into the hallway with red wall art. Walk through the entry into the kitchen area. Walk passed the counter tops and refrigerator and continue up to the rustic dining table ahead.

Walk out of the bathroom door into the hallway with red wall art. Walk through the entry into the kitchen area. Walk passed the counter tops and refrigerator and continue up to the rustic dining table ahead.

Fig. 4. The attention weights for all words in the instruction before and after using Action-Contextual Encoding Module. These weights are depicted using a gradient scale, where blue indicates lower attention values and red corresponds to higher values. For each example, the upper instruction represents the condition without the Action-Contextual Encoding Module, while the lower shows the condition with the Action-Contextual Encoding Module.

4.7 Qualitative Results

4.7.1 Visualization showing the effect produced by the Action-Contextual Encoding Module. As shown in Fig. 4, we present the attention A'_t of the agent to individual words in the instruction at a specific location. Before applying the Action-Contextual Encoding Module, the agent primarily focuses on words related to the current visual

Instruction: Walk forward into the next room and up the stairs. Stop at the top of the stairs. Go down the rest of the stairs and go through the doorway at the bottom. Walk straight through the room with the guitar painting and enter the room with some chairs. Stop after entering this room.

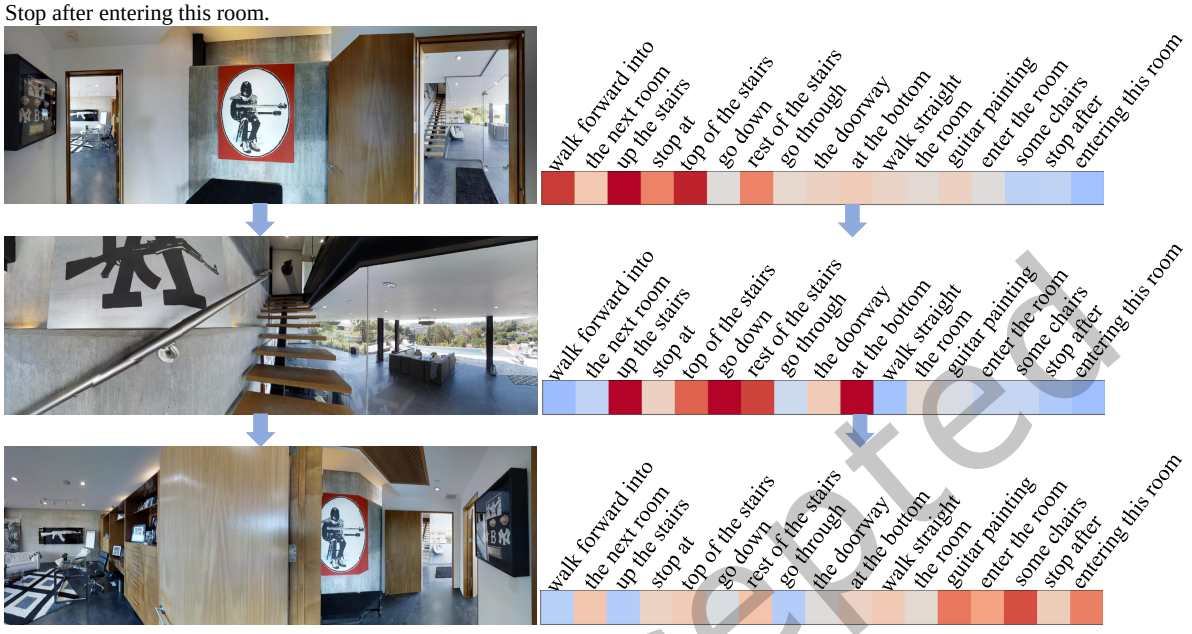


Fig. 5. The variation in how the agent pays attention to different words in the instruction as it moves through various positions. Here, the attention for a phase is determined by taking the average of the attention weights for all words within that phase. The attention is represented by a color scale ranging from blue to red, where blue indicates low attention and red indicates high attention.

information, such as room or object names like "stairs" or "door," while largely ignoring critical action words. This leads to a noticeable drop in navigation success rates. However, after applying the Action-Contextual Encoding Module, the agent begins to focus more on action-related words, such as "walk down" or "continue forward", effectively improving navigation accuracy. Additionally, we also observe that the agent also pays considerable attention to certain articles, conjunctions, and prepositions in the instruction. This is likely because these words appear alongside relevant actions and objects frequently.

4.7.2 Visualization showing the effect produced by the Dynamic Instruction Weighting Module. To more clearly demonstrate the function of the Dynamic Instruction Weighting Module, we visualized the agent's attention to different words in the instruction as it moves through various positions. As shown in Fig. 5, at the beginning of navigation, the agent primarily focuses on the first few words of the instruction. As the agent progresses, the Dynamic Instruction Weighting Module adjusts the weighting of words in the instruction based on features obtained at the current position. The agent gradually increases its attention to subsequent words, effectively reducing the interference from irrelevant words during navigation.

5 CONCLUSION

In this paper, to address the misalignment between visual observations and textual instructions in the long-instruction VLN task, we introduced the Action-Contextual Encoding Module and the Dynamic Instruction Weighting Module. The former ensured alignment between visual features and action-related words in the

instructions by incorporating action information into the visual observations. The latter continuously updated the weight of each word in the instruction, allowing the agent to focus primarily on those words relevant to its current position. Extensive experiments demonstrated that these two modules played a crucial role in improving performance on the long-instruction VLN task. However, we also observed that our model is not able to achieve perfect alignment between visual observations and textual instructions. Certain frequently occurring nouns tended to receive excessive attention from the agent, even when they were not relevant to the current visual context. In future work, we plan to further optimize the model to address this issue.

REFERENCES

[1] Dong An, Yuankai Qi, Yan Huang, Qi Wu, Liang Wang, and Tieniu Tan. 2021. Neighbor-view Enhanced Model for Vision and Language Navigation. In *Proceedings of the 29th ACM International Conference on Multimedia* (Virtual Event, China) *(MM '21)*. Association for Computing Machinery, New York, NY, USA, 5101–5109. https://doi.org/10.1145/3474085.3475282

[2] Dong An, Yuankai Qi, Yangguang Li, Yan Huang, Liang Wang, Tieniu Tan, and Jing Shao. 2022. BEVBert: Multimodal Map Pre-training for Language-guided Navigation. https://doi.org/10.48550/arXiv.2212.04385 arXiv:2212.04385

[3] Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. 2025. ETPNav: Evolving Topological Planning for Vision-Language Navigation in Continuous Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 47, 7 (2025), 5130–5145. https://doi.org/10.1109/TPAMI.2024.3386695

[4] Peter Anderson, Ayush Shrivastava, Devi Parikh, Dhruv Batra, and Stefan Lee. 2019. Chasing ghosts: instruction following as bayesian state tracking. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, Article 34, 11 pages.

[5] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3674–3683. https://doi.org/10.1109/CVPR.2018.00387

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D Data in Indoor Environments. In *2017 International Conference on 3D Vision (3DV)*. 667–676. https://doi.org/10.1109/3DV.2017.00081

[7] Jingwen Chen, Jianjie Luo, Yingwei Pan, Yehao Li, Ting Yao, Hongyang Chao, and Tao Mei. 2023. Boosting Vision-and-Language Navigation with Direction Guiding and Backtracing. *ACM Trans. Multimedia Comput. Commun. Appl.* 19, 1, Article 9 (Jan. 2023), 16 pages. https://doi.org/10.1145/3526024

[8] Jinyu Chen, Wenguan Wang, Si Liu, Hongsheng Li, and Yi Yang. 2023. Omnidirectional information gathering for knowledge transfer-based audio-visual navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10993–11003.

[9] Kevin Chen, Junshen K Chen, Jo Chuang, Marynel Vázquez, and Silvio Savarese. 2021. Topological planning with transformers for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11276–11286.

[10] Peihao Chen, Dongyu Ji, Kunyang Lin, Runhao Zeng, Thomas Li, Mingkui Tan, and Chuang Gan. 2022. Weakly-Supervised Multi-Granularity Map Learning for Vision-and-Language Navigation. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 38149–38161.

[11] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. 2021. History Aware Multimodal Transformer for Vision-and-Language Navigation. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 5834–5847.

[12] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 2022. Learning from unlabeled 3d environments for vision-and-language navigation. In *European Conference on Computer Vision*. Springer, 638–655.

[13] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 2022. Think Global, Act Local: Dual-scale Graph Transformer for Vision-and-Language Navigation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16516–16526. https://doi.org/10.1109/CVPR52688.2022.01604

[14] Ronghao Dang, Zhuofan Shi, Liuyi Wang, Zongtao He, Chengju Liu, and Qijun Chen. 2022. Unbiased directed object attention graph for object navigation. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3617–3627.

[15] Ronghao Dang, Liuyi Wang, Zongtao He, Shuai Su, Jiagui Tang, Chengju Liu, and Qijun Chen. 2023. Search for or Navigate to? Dual Adaptive Thinking for Object Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 8250–8259.

[16] Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. 2020. Evolving Graphical Planner: Contextual Global Planning for Vision-and-Language Navigation. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 20660–20672.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[18] Zi-Yi Dou and Nanyun Peng. 2022. FOAM: A Follower-aware Speaker Model For Vision-and-Language Navigation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 4332–4340. https://doi.org/10.18653/v1/2022.naacl-main.322

[19] Hugh Durrant-Whyte and Tim Bailey. 2006. Simultaneous localization and mapping: part I. *IEEE robotics & automation magazine* 13, 2 (2006), 99–110.

[20] A. Elfes. 2013. Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception. https://doi.org/10.48550/arXiv.1304.1098 arXiv:1304.1098

[21] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-Follower Models for Vision-and-Language Navigation. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc., 3318–3329.

[22] Chen Gao, Jinyu Chen, Si Liu, Luting Wang, Qiong Zhang, and Qi Wu. 2021. Room-and-object aware knowledge reasoning for remote embodied referring expression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3064–3073.

[23] Georgios Georgakis, Karl Schmeckpeper, Karan Wanchoo, Soham Dan, Eleni Miltsakaki, Dan Roth, and Kostas Daniilidis. 2022. Cross-modal map learning for vision and language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15460–15470.

[24] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. 2021. Airbert: In-Domain Pretraining for Vision-and-Language Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 1634–1643.

[25] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. 2020. Towards Learning a Generic Agent for Vision-and-Language Navigation via Pre-Training. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 13134–13143. https://doi.org/10.1109/CVPR42600.2020.01315

[26] Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. 2020. Language and Visual Entity Relationship Graph for Agent Navigation. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 7685–7696.

[27] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. 2021. VLN BERT: A Recurrent Vision-and-Language BERT for Navigation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1643–1653. https://doi.org/10.1109/CVPR46437.2021.00169

[28] Yicong Hong, Yang Zhou, Ruiyi Zhang, Franck Dernoncourt, Trung Bui, Stephen Gould, and Hao Tan. 2023. Learning navigational visual representations with semantic map supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3055–3067.

[29] Xiaobo Hu, Youfang Lin, Hehe Fan, Shuo Wang, Zhihao Wu, and Kai Lv. 2024. Building Category Graphs Representation with Spatial and Temporal Attention for Visual Navigation. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 7, Article 217 (May 2024), 22 pages. https://doi.org/10.1145/3653714

[30] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. 2019. General evaluation for instruction conditioned navigation using dynamic time warping. In *Visually Grounded Interaction and Language (ViGIL), NeurIPS 2019 Workshop*. https://vigilworkshop.github.io/static/papers-2019/33.pdf

[31] Muhammad Zubair Irshad, Niluthpol Chowdhury Mithun, Zachary Seymour, Han-Pang Chiu, Supun Samarasekera, and Rakesh Kumar. 2022. Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. In *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 4065–4071.

[32] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. 2019. Stay on the Path: Instruction Fidelity in Vision-and-Language Navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 1862–1872. https://doi.org/10.18653/v1/P19-1181

[33] Aishwarya Kamath, Peter Anderson, Su Wang, Jing Yu Koh, Alexander Ku, Austin Waters, Yinfei Yang, Jason Baldridge, and Zarana Parekh. 2023. A New Path: Scaling Vision-and-Language Navigation with Synthetic Instructions and Imitation Learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10813–10823. https://doi.org/10.1109/CVPR52729.2023.01041

[34] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=YicbFdNTTy

[35] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. 2020. Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 104–120.

[36] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*. 4392–4412. https://doi.org/10.18653/v1/2020.emnlp-main.356

[37] Federico Landi, Lorenzo Baraldi, Marcella Cornia, Massimiliano Corsini, and Rita Cucchiara. 2021. Multimodal attention networks for low-level vision-and-language navigation. *Computer Vision and Image Understanding* 210 (2021), 103255. https://doi.org/10.1016/j.cviu.2021.103255

[38] Jialu Li, Hao Tan, and Mohit Bansal. 2021. Improving Cross-Modal Alignment in Vision Language Navigation via Syntactic Information. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 1041–1050. https://doi.org/10.18653/v1/2021.naacl-main.82

[39] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A Smith, and Yejin Choi. 2019. Robust Navigation with Language Pretraining and Stochastic Sampling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 1494–1499.

[40] Xiangyang Li, Zihan Wang, Jiahao Yang, Yaowei Wang, and Shuqiang Jiang. 2023. KERM: Knowledge Enhanced Reasoning for Vision-and-Language Navigation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2583–2592. https://doi.org/10.1109/CVPR52729.2023.00254

[41] Bingqian Lin, Yi Zhu, Zicong Chen, Xiwen Liang, Jianzhuang Liu, and Xiaodan Liang. 2022. ADAPT: Vision-Language Navigation with Modality-Aligned Action Prompts. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15375–15385. https://doi.org/10.1109/CVPR52688.2022.01496

[42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 740–755.

[43] Chong Liu, Fengda Zhu, Xiaojun Chang, Xiaodan Liang, Zongyuan Ge, and Yi-Dong Shen. 2021. Vision-Language Navigation with Random Environmental Mixup. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 1624–1634. https://doi.org/10.1109/ICCV48922.2021.00167

[44] Rui Liu, Wenguan Wang, and Yi Yang. 2024. Volumetric Environment Representation for Vision-Language Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16317–16328.

[45] Rui Liu, Xiaohan Wang, Wenguan Wang, and Yi Yang. 2023. Bird's-Eye-View Scene Graph for Vision-Language Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10968–10980.

[46] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://doi.org/10.48550/arXiv.1907.11692 arXiv:1907.11692

[47] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*. 6732–6740.

[48] Bahram Mohammadi, Yicong Hong, Yuankai Qi, Qi Wu, Shirui Pan, and Javen Qinfeng Shi. 2024. Augmented Commonsense Knowledge for Remote Object Grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 4269–4277. https://doi.org/10.1609/aaai.v38i5.28223

[49] Abhinav Moudgil, Arjun Majumdar, Harsh Agrawal, Stefan Lee, and Dhruv Batra. 2021. SOAT: A Scene- and Object-Aware Transformer for Vision-and-Language Navigation. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 7357–7367.

[50] Abhinav Moudgil, Arjun Majumdar, Harsh Agrawal, Stefan Lee, and Dhruv Batra. 2021. SOAT: A Scene- and Object-Aware Transformer for Vision-and-Language Navigation. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 7357–7367.

[51] Alexander Pashevich, Cordelia Schmid, and Chen Sun. 2021. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 15942–15952.

[52] Yuankai Qi, Zizheng Pan, Yicong Hong, Ming-Hsuan Yang, Anton van den Hengel, and Qi Wu. 2021. The Road to Know-Where: An Object-and-Room Informed Sequential BERT for Indoor Vision-Language Navigation. In *ICCV*. 1655–1664.

[53] Yuankai Qi, Zizheng Pan, Shengping Zhang, Anton van den Hengel, and Qi Wu. 2020. Object-and-action aware model for visual language navigation. In *European Conference on Computer Vision*. Springer, 303–317.

[54] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020. REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9979–9988. https://doi.org/10.1109/CVPR42600.2020.01000

[55] Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. 2022. HOP: History-and-Order Aware Pretraining for Vision-and-Language Navigation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15397–15406. https://doi.org/10.1109/CVPR52688.2022.01498

[56] Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. 2023. HOP+: History-Enhanced and Order-Aware Pre-Training for Vision-and-Language Navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 7 (2023), 8524–8537. https://doi.org/10.1109/TPAMI.2023.3234243

[57] Yanyuan Qiao, Yuankai Qi, Zheng Yu, Jing Liu, and Qi Wu. 2023. March in Chat: Interactive Prompting for Remote Embodied Referring Expression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 15758–15767.

[58] Wenda Qin, Teruhisa Misu, and Derry Wijaya. 2021. Explore the Potential Performance of Vision-and-Language Navigation Model: A Snapshot Ensemble Method. https://doi.org/10.48550/arXiv.2111.14267 arXiv:2111.14267 [cs.CV]

[59] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[60] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6 (2017), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

[61] Akshita Maradapu Vera Venkata Sai, Chenyu Wang, Zhipeng Cai, and Yingshu Li. 2024. Navigating the Digital Twin Network landscape: A survey on architecture, applications, privacy and security. *High-Confidence Computing* 4, 4 (2024), 100269. https://doi.org/10.1016/j.hcc.2024.100269

[62] Hao Tan and Mohit Bansal. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

[63] Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2610–2621.

[64] Sinan Tan, Kuankuan Sima, Dunzheng Wang, Mengmeng Ge, Di Guo, and Huaping Liu. 2025. Self-Supervised 3-D Semantic Representation Learning for Vision-and-Language Navigation. *IEEE Transactions on Neural Networks and Learning Systems* 36, 4 (2025), 6738–6751. https://doi.org/10.1109/TNNLS.2024.3395633

[65] Hanqing Wang, Wenguan Wang, Wei Liang, Caiming Xiong, and Jianbing Shen. 2021. Structured scene memory for vision-language navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*. 8455–8464.

[66] Liuyi Wang, Zongtao He, Ronghao Dang, Mengjiao Shen, Chengju Liu, and Qijun Chen. 2024. Vision-and-Language Navigation via Causal Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13139–13150.

[67] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. 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*. 6629–6638.

[68] Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. 2023. LANA: A Language-Capable Navigator for Instruction Following and Generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 19048–19058. https://doi.org/10.1109/CVPR52729.2023.01826

[69] Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. 2023. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12009–12020.

[70] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. 2023. GridMM: Grid Memory Map for Vision-and-Language Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 15625–15636.

[71] Dongxiao Yu, Hengming Zhang, Yan Huang, and Zhenzhen Xie. 2025. Data distribution inference attack in federated learning via reinforcement learning support. *High-Confidence Computing* 5, 1 (2025), 100235. https://doi.org/10.1016/j.hcc.2024.100235

[72] Chongyang Zhao, Yuankai Qi, and Qi Wu. 2023. Mind the Gap: Improving Success Rate of Vision-and-Language Navigation by Revisiting Oracle Success Routes. In *Proceedings of the 31st ACM International Conference on Multimedia* (Ottawa ON, Canada) *(MM '23)*. Association for Computing Machinery, New York, NY, USA, 4349–4358. https://doi.org/10.1145/3581783.3611736

[73] Yusheng Zhao, Jinyu Chen, Chen Gao, Wenguan Wang, Lirong Yang, Haibing Ren, Huaxia Xia, and Si Liu. 2022. Target-Driven Structured Transformer Planner for Vision-Language Navigation. In *Proceedings of the 30th ACM International Conference on Multimedia*. 4194–4203.

[74] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. 2020. Vision-Language Navigation With Self-Supervised Auxiliary Reasoning Tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10009–10019. https://doi.org/10.1109/CVPR42600.2020.01003

[75] Wanrong Zhu, Yuankai Qi, Pradyumna Narayana, Kazoo Sone, Sugato Basu, Xin Wang, Qi Wu, Miguel Eckstein, and William Yang Wang. 2022. Diagnosing Vision-and-Language Navigation: What Really Matters. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 5981–5993.