2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

StratXplore: Strategic Novelty-seeking and Instruction-aligned Exploration for Vision and Language Navigation

Muraleekrishna Gopinathan, Jumana Abu-Khalaf, David Suter and Martin Masek

***Abstract*— Embodied navigation requires robots to understand and interact with the environment based on given tasks. Vision-Language Navigation (VLN) is an embodied navigation task, where a robot navigates within a previously seen and unseen environment, based on linguistic instruction and visual inputs. VLN agents need access to both local and global action spaces; former for immediate decision making and the latter for recovering from navigational mistakes. Prior VLN agents rely only on instruction-viewpoint alignment for local and global decision making and back-track to a previously visited viewpoint, if the instruction and its current viewpoint mismatches. These methods are prone to mistakes, due to the complexity of the instruction and partial observability of the environment. We posit that, back-tracking is sub-optimal and agent that is aware of its mistakes can recover efficiently. For optimal recovery, exploration should be extended to unexplored viewpoints (or frontiers). The optimal frontier is a recently observed but unexplored viewpoint that aligns with the instruction and is novel. We introduce a memory-based and mistake-aware path planning strategy for VLN agents, called *StratXplore*, that presents global and local action planning to select the optimal frontier for path correction. The proposed method collects all past actions and viewpoint features during navigation and then selects the optimal frontier suitable for recovery. Experimental results show this simple yet effective strategy improves the success rate on two VLN datasets with different task complexities.**

I. INTRODUCTION

Path planning and navigation in previously unseen environments is a challenging and widely studied problem in robotics. Vision-Language Navigation (VLN) is a robotic task that aims to impart language-conforming path planning capabilities in robots [1], [2]. Current state-of-the-art methods in VLN build and utilise topological representation of the environment for path planning. However, there is a significant gap between how humans and VLN agents navigate in unseen real-world environments [3], [4]. This is attributed to the diversity of the environment and the arbitrariness of human language. In particular, agents performing long-horizon language-following tasks can become perplexed in unseen environments and eventually make mistakes [5]. In this paper, we address the challenge of path planning in unseen environments and propose a novel strategy for VLN agents to recover from navigational mistakes.

Let us picture a real-world scenario, where a human is given an instruction to "... move forward, keeping the pictures to the left side, then enter the toilet ..." (Fig. 1).

The authors are with Centre for Artificial Intelligence and Machine Learning, School of Science, Edith Cowan University, 270 Joondalup Dr, Joondalup, WA 6027, Australia. {k.gopinathan, j.abukhalaf, m.masek, d.suter}@ecu.edu.au

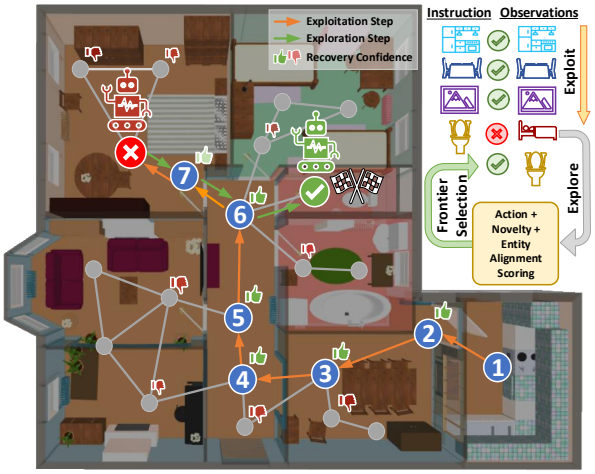

Fig. 1. **Overview**. StratXplore enables an embodied agent to correct its path by exploring frontiers that are both novel and conforms to the given instruction. Here, *exploit* refers to selecting one of the local candidate directions and *explore* considers all unexplored frontiers from the memory.

Although it may seem straightforward, this can potentially be ambiguous if, for example, the toilet is not visible from the current viewpoint 6. The human may end up entering the wrong room on the left instead (e.g. bedroom) ✗. Typically, when a human makes a navigational mistake, they backtrack and pursue another direction. Similarly, robots can face ambiguity in long-horizon navigational tasks where turn-by-turn instructions are unavailable or the environment is not fully observable. Therefore, in an open-vocabulary real-world setting, the performance of a robot will be heavily dependent on their path planning and error recovery [6].

Now a natural question is, *How can the agent recover from navigational mistakes?* Traditional strategies applied to object-search problems, suggest that *curiously* seeking novel viewpoints (novelty-seeking) can benefit error recovery and task success [7]. While enticing, directly applying this method is impractical in VLN task because of its strict need for instruction-path agreement. Hence, exploring for the sake of *curiosity* may result in the agent deviating form the correct path. Instead, hierarchical planners [8], [9], [10], [11] perform back-tracking using dual-scale planning; fine-scale for local planning and coarse-scale for back-tracking. The agent 'jumps' to a frontier (unexplored viewpoint) if the planner assigns a higher action probability to that frontier than to the current candidate directions. These methods have two main limitations; (1) their environment *state* representation, used for planning, is cluttered with previous correct and incorrect

visited viewpoints, suppressing the importance of optimal frontiers in decision making (2) these strategies allocate equal significance to all viewpoints, irrespective of how close they are to the goal. We aim to combat the limitations of a hierarchical planner.

The first issue can be addressed by strategically selecting *relevant* frontiers based on their novelty (amount of new information) and correspondence with the given instruction. The second issue can be tackled by prioritising temporally *recent* frontiers over the initial ones. We posit that the *recent* frontiers are more likely to be closer to the goal than initial frontiers making them more significant for recovery. Here, the final selection of potential frontiers is a set of all unexplored viewpoints ranked by decreasing order of *relevance* based on the recency of the observation.

Our strategic path planning is performed in two steps. Initially, the agent exploits the action decisions from a cross-modal action proposal module (introduced in §III) while also predicting the agent's confidence in candidate directions (depicted by 👍👎 in Fig. 1). When the agent learns that it made a mistake based on the confidence scores, it switches to the exploration mode. Here, the confidence score for a direction signifies the likelihood of an agent to reach an instruction-aligned path, if it were pursued. During exploration, StratXplore ranks *relevant* frontiers and selects the optimal frontier with the highest rank. Our agent navigates to the optimal frontier via the shortest path to correct the error. Finally, the agent switches to the exploitation mode.

Our contribution in this paper is as follows:

1) We propose a novel progress monitoring method that quantifies the agent's confidence in task conformity, if any of the candidate direction is pursued next (§III). This progress signal is simpler to estimate compared to existing methods.
2) We introduce a new exploration strategy to select the optimal frontier based on global and local landmark information for recovery. This is determined by the viewpoint temporal recency, viewpoint novelty and instruction-viewpoint correspondence (§IV). To the best of our knowledge, this is the first study in VLN on this front.
3) We propose an auxiliary learning task that trains the multi-modal planner to identify deviation from an instructed trajectory (§V-A).

II. Related Work

A. Path planning in VLN

Path planning is a crucial capability for any navigation agent [12], [13]. Conventional Vision-and-Language Navigation (VLN) agents are typically constrained to local action space, where choices are limited to the current candidate directions [14], [15]. Error recovery using these myopic strategies leads to repeated actions, requiring an agent to back-track via each visited step and re-evaluate them. To determine when to back-track, recent error recovery methods [16], [17] estimate the progress based on visited viewpoints and back-track if the progress is diminishing. AuxRN [18] used progress monitoring as a training objective instead. However, these sparse progress signals are harder to estimate during navigation in unseen environments, as the distance to the goal is unknown without pre-exploration. Additionally, another common limitation of these agents is that only visited viewpoints are stored in their memory, limiting the action space.

Transformer-based planners [9], [10], ameliorate this shortcoming by employing hierarchical decision making on global and local action spaces. The local and global decision contexts are generated independently from two cross-modal transformers and fused later for action prediction. Late fusion makes the global planner oblivious to local planner's decisions and vice versa. Furthermore, this fusion method fails to take advantage of past action scores and the significance of recent observations over initial ones. Our method incentives the agent to select recent viewpoints that align with the instruction both globally and locally.

B. Memory representations in VLN

Memory in robotics encompasses the environmental and navigational *state* representations of an agent which can be used for localisation, scene understanding, question answering, object discovery and instruction following [5], [19], [20]. Path planning in a fully explored environment can be reduced to a shortest path problem [3] and an abstract-level memory (comprising of low-detail scene features) is sufficient for the agent. In contrast, a more detailed memory structure is crucial for an agent navigating in unseen environments. Inspired from human memory, robotic memory evolved from storing metric [21] to semantic [10] information in the form of recurrent [22], topological [9], hierarchical [14], and topo-metric [10] representations. Notable works in visual navigation, VGM [20] and WGM [23], encode visual graph memory comprising of scene features, but the memory is used solely for localising the agent. This method cannot be directly adopted for the VLN task where the exploration scheme should guarantee instruction-trajectory correspondence. Methods that use memory for path planning such as SSM [5], store viewpoints from correct and incorrect actions in the memory. A disproportionate amount of incorrect actions can cause the transformer-based planner to pay attention to incorrect viewpoints, causing the planner to make sub-optimal decisions. Instead, we propose curating the viewpoint features that are added to the memory dependent on their local relevance, temporal-importance, and novelty. Our method, which also uses a graph memory, employs a buffer that accumulates object-centric features from viewpoints. This is used for identifying unique viewpoints (for novelty) and comparing instruction-viewpoint correspondence with other frontier viewpoints.

III. Our Approach

In this section, we introduce the general VLN problem and our strategic exploration scheme for language-guided

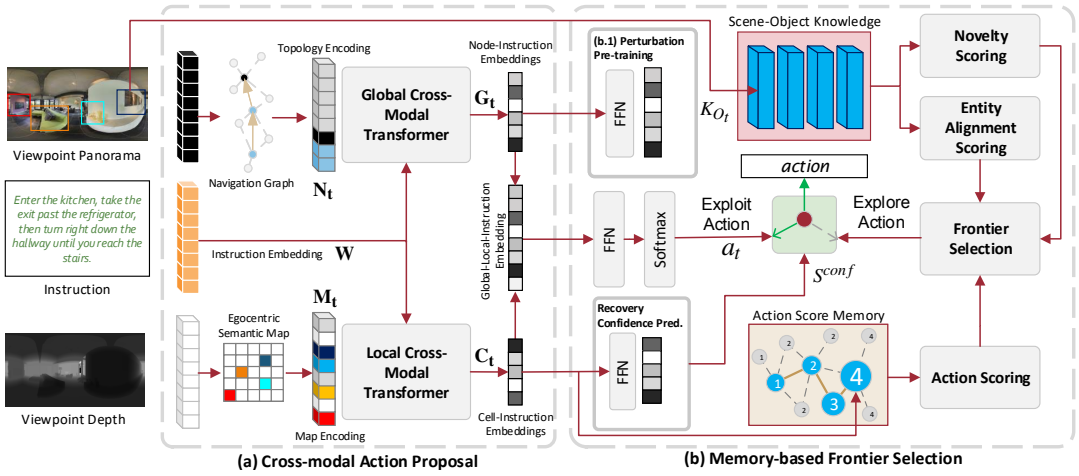

Fig. 2. **Model Architecture of StratXplore.** (a) Fused action proposal from Global and Local Cross-modal transformers (CMT) is used for exploitation (b) When the recovery confidence S^{conf} of current candidates drops below a threshold c_{thresh}, the agent chooses to explore. Frontier selector considers the optimal recent-and-novel and instruction-aligned frontier to explore. Blocks are hyperlinked to relevant sections.

navigation agents, called StratXplore. The architecture of our method is depicted in Fig. 2.

A. Task Definition

The Vision-and-Language Navigation (VLN) task integrates natural language processing and visual perception in a pre-defined graph based environment. In VLN, an agent starts at an initial location within a new indoor environment and follows language instructions to reach the goal. The agent does not have access to the full environment graph during the navigation. At each time step, the agent observes a panoramic viewpoint which comprises 36 single views. The navigable subset of these views is referred to as *candidate* directions. The agent selects one candidate direction to move to the next viewpoint and this process repeats until the agent decides to stop. An ideal agent strictly follows the instruction and stops within 3 meters of the goal location.

B. Inputs

The inputs to the agent are instruction, panoramic viewpoint images, panoramic depth maps, and the agent's world poses (location and orientation) at each step. We use these inputs to generate the following modality encodings.

1) Instruction Encoding: The word tokens from the instruction are embedded using an embedding layer. The resulting embedding is summed with the token position embedding and fed to a multi-layer language transformer to obtain the contextual instruction representation $\mathbf{W}$.

2) Topology Encoding: The panoramic image $\mathbf{O_t}$ of the viewpoint node, observed at each step, is applied to a vision transformer to obtain a viewpoint feature. The pose embedding is $[cos(\theta), \sin(\theta), \cos(\phi), \sin(\phi)]$, where θ and ϕ are relative heading and elevation, respectively. The node embedding of the viewpoint is integrated into the navigation memory, by adding the viewpoint features, pose embedding of the step, and step index. A special stop direction is also included to indicate a stop action. The node embeddings of the entire graph $\mathbf{N_t}$ and the encoded instruction $\mathbf{W}$ are applied to a multi-layer cross-modal transformer as shown in Fig. 2. The cross-attention and graph-aware self-attention (GASA) [9] in the transformer, model the language-viewpoint inter-modal relationships. GASA considers node embeddings and viewpoint adjacency to calculate global node-instruction embedding, $\mathbf{G_t}$. This is used for the exploitation action proposal.

3) Ego-centric Semantic Map Encoding: Local planning requires knowing which direction is to be pursued next. For this, we devise the local action space as follows. At first, the semantic features of the viewpoint are obtained using an object detector. Then, the ego-centric (polar) semantic-metric map is obtained by inverse-projecting the semantic features from the image space to the world space using the viewpoint's depth map. Technically, this is done by shooting rays from the camera centre to the semantic feature using the depth value of each pixel (lift) and projecting each feature onto the world ground plane (splat) following [10]. Each cell of the Birds-Eye-View (BEV) map represents a region of ground plane and contains average-pooled semantic features of that area. Finally, the map encoding $\mathbf{M_t}$ is obtained as the cell-wise sum of the polar feature map, navigability features (signifying occlusions/obstructions) and polar position embeddings of the grid. To capture instruction-viewpoint alignment useful for local decision making, we use a cross-modal (local) transformer with cross and self-attention and obtain the cell-instruction contextual representation, $\mathbf{C_t}$.

C. Action Proposal

For the exploitation action proposal, we follow previous methods [9], [10] for fusing the global and local contextual embeddings from the respective transformers (CMT) to obtain the global action proposal. At each step t, the action scores for each viewpoints are obtained as,

$$\mathbf{G_t} = \text{CMT}_{global}(\mathbf{W}, \mathbf{N_t}) \quad (1)$$

$$s_g = \text{FFN}_g(\mathbf{G_t}) \quad (2)$$

$$\mathbf{C_t} = \text{CMT}_{local}(\mathbf{W}, \mathbf{M_t}) \quad (3)$$

$$s_l = \text{FFN}(\mathbf{C_t}) \quad (4)$$

$$a_t = arg\max\ p([s_g; s_l]\mathbf{W}) \quad (5)$$

where FFNs are feed forward neural networks that predict action scores for global and local contexts. The action proposal selects the node with the highest exploit action probability a_t after fusing (represented by $[;]$) the global s_g and local s_l action features.

This proposal is prone to navigational mistakes due to task complexity and agent needs to identify and recover from it. Because the network weights are not shared between the both CMTs, contextual representation generated by one does not affect the other. Hence, the object landmark detections are localised to the local map encoder and place landmarks are localised to global planner. This hinders effective error correction. To alleviate this, StratXplore considers both global and local landmark information for recovery.

1) Detecting a navigation mistake: The agent needs to have an implicit notion of navigational progress for successful navigation. For this, we propose two training schemes aimed at imparting deviation awareness and progress monitoring abilities to the planner. Firstly, we propose offline pre-training of the global cross-modal transformer (§V-A) to detect agent deviating from the optimal path (Fig. 2 (b.1)). Secondly, we use a predictor to estimate a *confidence score* S^{conf} during navigation. If the score for each of the candidate directions is less than a threshold c_{thresh}, the agent chooses to explore, otherwise it continues to exploit. Unlike existing self-monitoring agents which estimate the navigation progress by training an neural network to predict the distance to goal location, this module estimates the likelihood of recovering to the optimal path for a candidate viewpoint chosen by the agent. This is a fine-grained mistake estimation signal predicted from the cell-instruction embedding $\mathbf{C_t}$ as follows

$$S^{conf} = sigmoid(\text{FFN}_c(\mathbf{C_t})) \tag{6}$$

The training process is explained in detail in §V-B.1.

2) Recovery: We hypothesise that the local action scores, novelty and task-conformity (instruction-viewpoint correspondence) assigned to all frontiers are important for assessing *relevant* frontiers for recovery. For instance, consider a frontier node, observed from various neighbouring viewpoints, obtains the relatively high action score from these observations. Logically, this direction (or frontier) is a good candidate as a *relevant* frontier based on its cumulative action score assigned from independent observations. In addition, exploring novel frontiers can yield unique information about the environment. These aspects enhance the agent's ability to identify the optimal frontier. Four scores are used to rank the candidates, namely, action proposal score S^{act}, the novelty of the viewpoint S^{novel}, the alignment of the viewpoint to the instruction S^{align}, and the recency of the viewpoint $S^{recency}$. We explain them in detail in the following section.

IV. Action-and-Knowledge based Frontier Selection

In this section, we explain the implementation of path correction and frontier selection. A frontier has two-levels of *relevance* - global and local. The local relevance is same as the action scores allocated by the Action Proposal module during the exploitation phase. This ensures that the scores relevant to any viewpoint at the time it is observed, are used for decision making. Global relevance is governed by temporal-recency to the current node as well as task-conformity. In effect, during frontier selection, local and global relevance are together considered for frontier ranking. To realise this, we use two memories namely: an Action memory (local relevance) and a Scene-Object memory dealing with the task-conformity and novelty (global relevance).

A. Action Memory

The action memory is a directed graph $G_{act} =< V_{t_{obs}} \varepsilon_{score} >$ where $V_{t_{obs}}$ represents both visited viewpoints and frontiers and ε_{score} represents viewpoint adjacency and their action score proposed by the planner. Each $V_{t_{obs}}$ includes three attributes: a viewpoint identifier, the latest observation time step t_{obs} and a visitation flag. The time step represents the order of visitation or observation i.e. both the visited viewpoint and its observed neighbours have the same time step $t = t_{obs}$. The $flag$ indicates if the node has been `visited` or not (i.e. `frontier`). The edges ε_{score} store the action score predicted by the action proposal module at each step. During exploitation, the edges that connect the visited nodes are set to 0, to prevent re-visitation. Note that action scores are normalised and can be considered as probabilities only in the neighbourhood but not in the overall graph context. Action scoring can be summarised as follows.

At time step t, the agent observes the environment and obtains a viewpoint, its neighbours and their connectivity. The current node obtains ($flag =$`visited`, $t_{obs} = t$) and the neighbours obtain ($flag =$`frontier`, $t_{obs} = t$). The edges ε_{score} of the neighbours are updated based on the action scores predicted by the action proposal module.

B. Scene-object Memory

Our scene-object memory provides additional viewpoint information to the planner in order to compare frontiers and make global decisions. Knowledge of relevant objects in a viewpoint assists the agent in strictly complying with the instruction. It is also useful for selecting novel frontiers that are different from the visited locations to prevent repeated actions. We develop a scene-object memory for this purpose.

The Scene-Object memory (Fig. 3) is a buffer accumulating object-related knowledge from viewpoints. This knowledge vector of a viewpoint is added to the memory only if it is novel (i.e. represents unique objects) with respect to other viewpoint knowledge vectors in the memory. Note that here the measure of uniqueness is object-centric, and hence high similarity between knowledge vectors of two viewpoints in the memory means they have more or less the same type of objects.

The knowledge vector is constructed for each viewpoint as follows. For each direction of the viewpoint, we select the top-K high confidence objects detected by an object detection model. We use the Faster R-CNN model [24] specifically trained on the Visual Genome dataset [25]. To represent

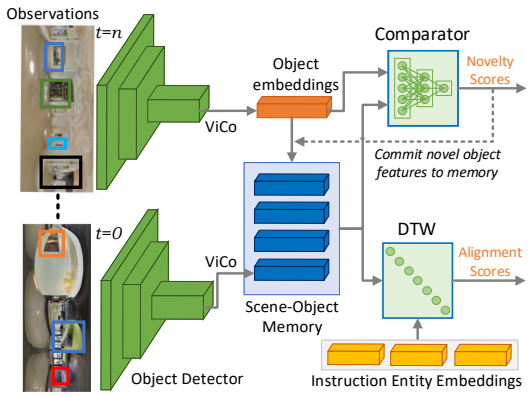

Fig. 3. **Novelty and Alignment Scoring.** At each time step, object features are added to memory if the objects are novel. During exploration both scores are used to rank frontiers.

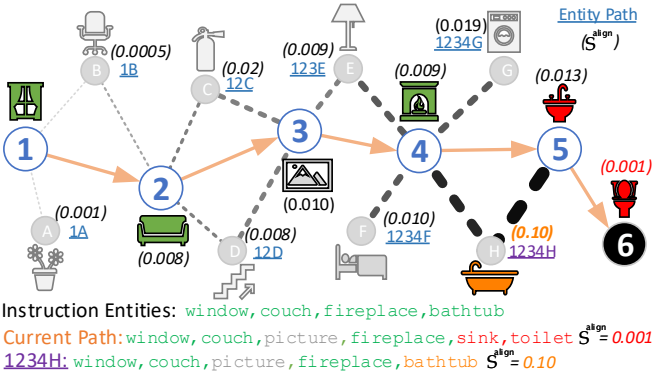

Fig. 4. **Example for Temporal prioritisation and Entity Alignment Scoring.** Recent memories (thicker connections) are prioritised over past actions to encourage exploration. S^{align} (values in parenthesis) is the highest for the path to the optimal frontier (here 1234H).

the objects' *knowledge* for a viewpoint, the ViCo [26] word embeddings of the object names are used. ViCo embeddings have visual and textual co-occurrence awareness i.e. objects that are seen together in the real world (chair and table) as well as in text (like in GloVe embeddings), are close in embedding space. In order to summarise the viewpoint objects, the sum of all object embeddings are used. As the frontiers are only partially observed from different angles via candidate directions of visited viewpoints, the frontier knowledge is the sum of object features of the respective directions. Contrastingly, the knowledge of the visited viewpoint K_{v_i} is the sum of object features of non-candidate directions. Next, we explain our method for measuring novelty and entity alignment.

C. Frontier Scoring

1) Novelty Scoring: At each time step t, the navigation graph is updated with unique knowledge of the node at that step. For the current node, the novelty score is the inverse of the cosine similarity between the viewpoint knowledge and all the elements in the memory:

$$S^{novel}_{\mathbf{O}_t} = \frac{\|K_{\mathbf{O}_t}\| \left\|\sum K^{mem \setminus \mathbf{O}_t}\right\|}{(K_{\mathbf{O}_t} \cdot \sum K^{mem \setminus \mathbf{O}_t})} \tag{7}$$

The knowledge is added to memory if $S^{novel}_{\mathbf{O}_t} > 2$, to signify largely different viewpoint objects.

2) Entity alignment scoring: Unlike the language-vision correspondence measured by the cross-modal transformer, this alignment score considers landmark words or entities (rooms, objects etc.) extracted from the instruction and compares it against the elements in the memory (Fig. 4). To ensure the unexplored viewpoint monotonically aligns with the instruction, we measure the Dynamic Time Warping (DTW) [27] cost between the entity sequence from the instruction and the knowledge vectors corresponding to viewpoints of the test path. A test path is a sequence of viewpoints that leads to a frontier. First to compute DTW cost, ViCo embedded entities are extracted from the instruction $K^{entities} = ViCo(\texttt{entities})$ and the knowledge vectors are extracted from the Scene-Object memory corresponding to test path. To reduce the computation cost in case of a large number of test paths, we filter frontiers with $S^{act} > 0.5$ and construct the test path as the shortest path $T_{\mathbf{O}_i}$ from the earliest visited viewpoints in the agent's trajectory, to every frontier viewpoint. The DTW score between the $K^{entities}$ and the knowledge sequence of the test paths in $T_{\mathbf{O}_i}$ (eg. the path 1234H in the figure). We use a computationally less expensive but accurate implementation for small sequences, FastDTW [28], with the euclidean distance as the distance function. The DTW cost is normalised to obtain the alignment score:

$$S^{align}_{\mathbf{O}_t} = exp\left(-\frac{DTW(K^T_{\mathbf{O}_i}, K^{entities})}{|K^T_{\mathbf{O}_i}||K^{entities}|}\right) \tag{8}$$

D. Frontier Selection

In the exploration mode, the memory is queried to find the optimal frontier. For this, the frontier selector queries the memory with all the un-normalised action scores. We rank the nodes as follows:

1) Select frontier nodes from action memory i.e. $flag = \texttt{frontier}$ and $t_{obs} < t$.
2) Score accumulation: Action scores assigned to incident edges of each frontier node ε_{score} are summed to obtain S^{act} of that frontier. This ensures frontiers that are relevant locally, is also relevant globally.
3) Temporal relevance calculation: We obtain the recency score $S^{recency} = \exp(\gamma(t_i - t))$ where the scores of nodes from recent history remain the same while scores from early steps are diminished based on decay factor γ (Fig. 4).
4) Obtain the novelty score S^{novel} for each unexplored location with respect to the current viewpoint.
5) Final scores for frontiers are obtained as $S = S^{act} * S^{recency}(S^{novel} + S^{align})$. Scores are normalised.

The node with the maximum score after normalisation is the optimal frontier. The agent executes the shortest path through the navigation graph to reach the node.

V. Training

A. Pre-training

Previous studies have applied different pre-training tasks to improve task generalisation in VLN. Pre-training provides

a holistic understanding of the task to the cross-modal planner and is a good starting point for downstream navigation models [10], [29]. Accordingly, we begin by pre-training our model with behaviour cloning based on offline expert demonstrations, alongside various vision-and-language related tasks. These tasks include masked language modelling (MLM), masked region classification (MRC), single-step action prediction (SAP), and object grounding (OG).

In order to train cross-modal transformers to identify deviation from the ground truth path, we introduce a deviation prediction (DP) task that predicts if parts of the agent trajectory sequence have deviated from the instruction path. The global node-language embedding $\mathbf{G_t}$ is fed to a 2 layer FFN and trained together with the aforementioned auxiliary tasks. This auxiliary training requires synthetic path demonstrations, which are derived from the R2R dataset. For this, we introduce carefully controlled perturbations to the ground truth paths. The perturbed path is comprised of parts of the ground truth path and a detour path commencing from a random viewpoint. The detour may end at: a location previously traversed in the ground truth path, one of the frontiers, or a random location in the vicinity (but not neighbours) of the ground truth path. The detoured segment is reconnected back with the remainder of the ground truth path via the shortest path.

The viewpoints of the resulting path of length N are labelled as on track (0), deviated (1), or recovering (2) i.e. $lab_{i1}^{N} \in \{0,1,2\}$. The classifier is optimised using cross entropy loss.

B. Navigation Training

We train the agent using imitation learning (IL). In IL, a teacher model suggests the next action based on the ground truth path and the action prediction is optimised using cross entropy loss.

1) Recovery Confidence Prediction: The recovery confidence prediction model is trained online using teacher-forcing. The nearest ground truth viewpoint for each candidate direction of the current viewpoint is calculated during navigation training. The confidence score, predicted by the model $p_{c,t}^{rec}$ for each candidate c, signifies the likelihood of recovery. The training target $y_{c,t}^{rec}$ is the normalised distance d_c from each candidate viewpoint c to the nearest location in the ground truth path i.e. $d_c = min(d_{c,GT})/d_{max}$. The target will be 1 if the agent is already on the ground truth path and $(1-d_c)$ as it deviates from it. The objective is to minimise the mean squared error (MSE) between the target and predicted confidence scores,

$$\mathscr{L}_{rcr} = \sum_{t=1}^{T} (y_{c,t}^{rec} - p_{c,t}^{rec})^2 \tag{9}$$

2) Action prediction: The action selection objective is optimised with a cross-entropy loss. The overall loss is the weighed sum of action prediction and recovery confidence,

$$\mathscr{L}_{loss} = -\lambda \sum_{t=1}^{T} (y_{c,t}^{nv} \log(p_{c,t}) - (1-\lambda)\mathscr{L}_{rcr} \tag{10}$$

where $p_{c,t}$ is the action probability of thecandidate direction c in viewpoint at step t, $y_{c,t}^{nv} \in \{0,1\}$ indicates ground-truth action, $\lambda = 0.4$ is the weight balancing the two losses.

VI. Experiments

We evaluate StarXplore using two VLN datasets for testing its room-finding capabilities, namely Room-to-Room (R2R) [2], Room-for-Room (R4R) and [30]. R2R has short turn-by-turn instructions, and R4R has coarser instructions and longer trajectories extended from R2R.

A. Implementation Details

We adopt BEVBert [10], a topology-learning transformer navigator that uses Birds-Eye-View map for local action prediction, as our baseline. We extend this model with our deviation pretraining, recovery confidence prediction and frontier selection, to impart error recovery capability. The auxiliary tasks (§V-A) used to pre-train the CMTs are mixed using the ratio MLM:SAP:MRC:OG:DP = 5:5:1:1:1:1. The hyperparameters for the baseline are set according to the model published in [10].

The action score memory has the temporal priority factor γ set to 0.1. The local BEV map size is set to 21 grids with 0.5m resolution based on the recommendation from the original work.

B. Evaluation Metrics

For evaluating the model's performance on R2R [2], four widely recognised metrics are employed: Trajectory Length (TL), Navigation Error (NE), Success Rate (SR) and Success Rate weighted by Path Length (SPL). NE is the distance from the goal to the agent's stopping position and SR assesses the frequency of successfully reaching the goal within 3m.

Additionally, in R4R [30], three other metrics are adopted as per existing studies: Coverage weighted by Length Score (CLS), Normalised Dynamic Time Warping (nDTW) [27], and Success rate weighted by normalized Dynamic Time Warping (SDTW), further expanding the evaluation framework.

VII. Results

A. R2R dataset

Table I compares the performance of StratXplore with current methods on the R2R task. The results displayed in the last row demonstrate that our model outperforms existing methods on SR and SPL across dataset split. Compared to our baseline, BEVBert [10], we can observe that the proposed model obtains relative improvement in SR (3.91%) and SPL (2.79%) in Test Unseen split. In particular, our method outperforms other explicit memory models such as SSM [5] (SR: +14.86, SPL: +18.79) and DUET [9] (SR: +6.89, SPL: +5.79) by an absolute margin. Similar improvement is seen when compared to progress monitoring methods such as [17], [18], [32], underlining the effectiveness of our proposed method.

TABLE I

QUALITATIVE COMPARISON (§VII-A) WITH STATE-OF-THE-ART METHODS ON R2R DATASET.

Methods	**Val Seen**				**Val Unseen**				**Test Unseen**			
	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑	TL	NE↓
Random	16	-	9.58	9.45	16	-	-	9.23	13	12	9.89	9.79
Human [31]	-	-	-	-	-	-	-	-	86	76	11.90	1.16
Seq2Seq [2]	6.0	39	11.33	-	22	-	**8.39**	7.84	20	18	**8.13**	7.85
VLN⟳BERT [15]	72	68	**11.13**	2.90	63	57	12.01	3.93	63	57	12.35	4.09
Self-monitoring† [17]	69	63	11.69	3.31	47	41	12.61	5.48	61	56	-	4.48
Regretful-Agent [32]	69	63	-	3.23	50	41	-	5.32	48	40	-	5.69
AuxRN [18]	70	67	-	3.33	55	50	-	5.28	55	51	-	5.15
HAMT [14]	76	72	11.15	2.51	66	61	11.46	**2.29**	65	60	12.27	3.93
SSM [5]	71	62	14.7	3.10	62	45	20.7	4.32	61	46	20.4	4.57
DUET [9]	79	73	12.32	2.28	72	60	13.94	3.31	69	59	14.73	3.65
BEVBert [10]	-	-	-	-	75	64	-	2.81	73	62	-	3.13
StratXplore (Ours)	**80.2**	**75.4**	12.16	**2.47**	**77.61**	**66.92**	12.94	**2.93**	**75.86**	**64.79**	14.36	**3.04**

TABLE II

QUALITATIVE COMPARISON (§VII-B) WITH THE STATE-OF-THE-ART METHODS ON R4R DATASET.

Methods	**Val Seen**						**Val Unseen**					
	NE↓	TL↓	SR↑	CLS↑	nDTW↑	SDTW↑	NE↓	TL↓	SR↑	CLS↑	nDTW↑	SDTW↑
Speaker-Follower [31]	5.35	15.4	52	0.46	-	-	8.47	19.9	24	0.30	-	-
RCM [33]	5.37	18.8	53	0.55	-	-	8.08	28.5	26	0.35	0.30	0.13
PTA (high-level) [34]	**4.54**	16.5	58	0.60	0.58	0.41	8.25	**17.7**	24	0.37	0.32	0.10
EGP [35]	-	-	-	-	-	-	**8.00**	18.3	30	0.44	0.37	0.18
E-Drop [36]	-	19.9	52	0.53	-	0.27	-	27.0	29	0.34	-	0.09
OAAM [37]	-	**11.8**	56	0.54	-	0.32	-	13.8	31	0.40	-	0.11
BabyWalk [38]	-	-	-	-	-	-	8.20	19.0	27	0.49	0.39	0.18
EntityGraph [39]	5.31	-	52	0.55	0.62	0.50	7.43	-	36	0.41	**0.47**	**0.34**
SSM [5]	4.60	19.4	63	0.65	0.56	0.44	8.27	22.1	32	0.53	0.39	0.19
StratXplore (Ours)	**5.26**	20.35	**66**	**0.67**	0.58	**0.46**	8.10	21.64	**38**	**0.55**	0.45	0.17

B. R4R dataset

Table II compares our model to other VLN methods on the R4R task. Our model shows the best success rate in both ValSeen (66%) and ValUnseen (35%) splits. StratXplore shows a relative improvement of 3. 17% and 3. 07% on SR and CLS in Val Seen split compared to SSM, respectively. Similarly, we see a relative improvement of 5. 55% (SR) and 3. 77% (CLS) on the Val Unseen split. This clearly demonstrates the impact of our method on long-horizon path planning tasks. However, the path correction has adversely impacted the TL and trajectory-shape-dependent scores such as nDTW and sDTW, nonetheless, the results are still comparable to those of the prior agents.

C. Qualitative comparison

We sample an interesting scenario from R2R ValUnseen split (Fig. 5). While both the baseline and our StratXplore agent make mistakes, StratXplore identifies the navigational mistake and recovers to the optimal frontier while the baseline agent fails to do so. Interestingly, deviation prediction training helps the agent identify the correct left turn.

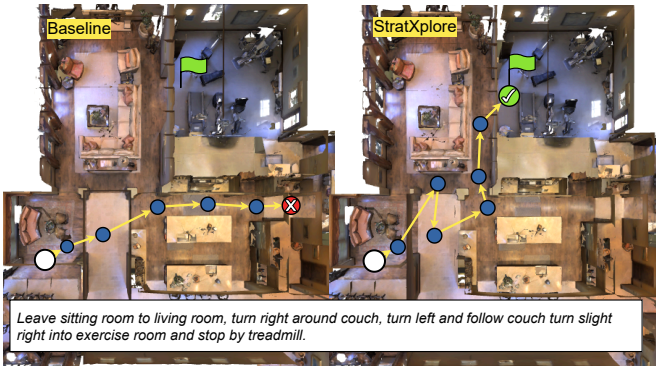

Fig. 5. **Qualitative comparison between trajectories of the baseline and StratXplore agents.** The baseline agent does not recover from the navigational mistake and continues to exploit the same direction and eventually fails. Although StratXplore makes a mistake by entering the second living room, it quickly corrects itself by moving to the best observed frontier (hallway).

D. Perturbation Study

We study the agent's ability to recover by *kidnapping* it to various viewpoints in the environment and measuring the change in navigational success (Table III). For this we devise 4 kidnapping scenarios with an increasing order of difficulty, i.e. kidnapping: 1) to a previously visited location (Visited), 2) to a location from the ground truth trajectory (Guiding), 3) toa 3-hop neighbourhood of the current path (3-Neighbourhood), and 4) to a random location close to the trajectory (Close). The first two scenarios measure the instruction-trajectory co-grounding ability while the latter two measure recover-ability of the planner during a critical failure. We test the performance of different memory types of the respective representative models: Recurrent (VLN⟳BERT [22]), Hierarchical (VLN-HAMT [14]), Topological (DUET [9]), Topo-metric (BEVBert [10]) and Dual Action-Novelty (Ours). The recurrent memory has the largest drop in SR (-9,-6,-12,-30) and SPL (-17,-11,-17,-36) in

TABLE III
CHANGE IN SUCCESS RATES AFTER KIDNAPPING AGENTS WITH DIFFERENT MEMORY TYPES

Memory	Baseline (SR,SPL)	Change in Success Rate (ΔSR,ΔSPL) Visited	Guiding	3-Neighbourhood	Close
#1 Recurrent	(63,57)	-9,-17	-6,-11	-12,-17	-30,-36
#2 Hierarchical	(66,61)	-9,-14	-7,-11	-8,-10	-24,-30
#3 Topological	(72,60)	-7,-10	+2,+1	-3,-7	-22,-25
#4 Topo-metric	(75,64)	-5,-7	+1,+1	-4,-4	-18,-21
#5 Ours	(76,65)	-4,-6	+3,+2	-2,-3	-13,-17

all kidnapping scenarios, revealing the inefficacy of short term memories in path recovery. Topological and Topo-metric memories demonstrate better recovery compared to methods #1 and #2. It is interesting to see that only the topological memories and our methods (Ours) leveraged guidance towards the goal. Also, in methods #1-#4 the success rates drop considerably even for visited viewpoints. One explanation is that the agent continues to repeat previous actions and fails to recover toward the goal. In contrast, our method identifies novel environments to navigate and recovers to novel frontiers in recent history. This allows the agent to identify the location it is kidnapped to and continue the navigation from one of the candidate frontiers. Hence, StratXplore demonstrates the lowest change in success rates ((4,-6),(+3,+2),(-2,-3),(-13,-17)) among all existing memory types used in VLN and can also leverage the movement towards the goal.

VIII. CONCLUSION

In this paper, we introduce a strategic exploration model, called StratXplore, designed to tackle the challenges encountered by Vision-Language-Navigation agents. We recognise error recovery as an essential capability of an embodied robot navigating in unseen or novel environments. Our method imparts four important aspects to VLN agents for error recovery: progress monitoring, ensuring task-conformity, seeking novel viewpoints and identifying a viewpoint's temporal-importance. Experimental results on R2R and R4R tasks demonstrate that our method is effective in improving navigational success in unseen environments.

Limitations and future work StratXplore agent rely on post-facto comparison of frontiers after a navigation mistake. To improve implicit awareness of a mistake and reduce added ranking cost, this method could be integrated with the planner rather than a separate error recovery module. This will be addressed in future work.

REFERENCES

[1] A. S. Huang, S. Tellex, A. Bachrach, *et al.*, "Natural language command of an autonomous micro-air vehicle," in *IROS*, 2010.

[2] P. Anderson, Q. Wu, D. Teney, *et al.*, "Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments," in *CVPR*, 2018.

[3] G. A. Sigurdsson, J. Thomason, G. S. Sukhatme, *et al.*, "RREx-BoT: Remote Referring Expressions with a Bag of Tricks," in *IROS*, 2023.

[4] M. Gopinathan, M. Masek, J. Abu-Khalaf, *et al.*, "Spatially-aware speaker for vision-and-language navigation instruction generation," in *ACL*, 2024.

[5] H. Wang, W. Wang, W. Liang, *et al.*, "Structured Scene Memory for Vision-Language Navigation," in *CVPR*, 2021.

[6] B. Chen, F. Xia, B. Ichter, *et al.*, "Open-vocabulary queryable scene representations for real world planning," in *ICRA*, 2023.

[7] D. Pathak, P. Agrawal, A. A. Efros, *et al.*, "Curiosity-driven exploration by self-supervised prediction," in *ICML*, 2017.

[8] M. Z. Irshad, C. Y. Ma, and Z. Kira, "Hierarchical Cross-Modal Agent for Robotics Vision-and-Language Navigation," *ICRA*, 2021.

[9] S. Chen, P.-L. Guhur, M. Tapaswi, *et al.*, "Think Global, Act Local: Dual-scale Graph Transformer for Vision-and-Language Navigation," in *CVPR*, 2022.

[10] D. An, Y. Qi, Y. Li, *et al.*, "BEVBert: Topo-Metric Map Pre-training for Language-guided Navigation," in *CVPR*, 2022.

[11] M. Hwang, J. Jeong, M. Kim, *et al.*, "Meta-Explore: Exploratory Hierarchical Vision-and-Language Navigation Using Scene Object Spectrum Grounding," in *CVPR*, 2023.

[12] H. Luo, A. Yue, Z.-W. Hong, *et al.*, "Stubborn: A strong baseline for indoor object navigation," in *IROS*, 2022.

[13] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Comm. ACM*, 1979.

[14] S. Chen, P.-L. Guhur, C. Schmid, *et al.*, "History Aware Multimodal Transformer for Vision-and-Language Navigation," in *NeurIPS*, 2021.

[15] Y. Hong, Q. Wu, Y. Qi, *et al.*, "VLNBert: A Recurrent Vision-and-Language BERT for Navigation," in *CVPR*, 2021.

[16] M. Zhu, B. Zhao, and T. Kong, "Navigating to Objects in Unseen Environments by Distance Prediction," *IROS*, 2022.

[17] C. Y. Ma, J. Lu, Z. Wu, *et al.*, "Self-monitoring navigation agent via auxiliary progress estimation," in *ICLR*, 2019.

[18] F. Zhu, Y. Zhu, X. Chang, *et al.*, "Vision-language navigation with self-supervised auxiliary reasoning tasks," in *CVPR*, 2020.

[19] M. Gopinathan, G. Truong, and J. Abu-Khalaf, "Indoor semantic scene understanding using 2d-3d fusion," in *DICTA*, 2021.

[20] O. Kwon, N. Kim, Y. Choi, *et al.*, "Visual graph memory with unsupervised representation for visual navigation," in *ICCV*, 2021.

[21] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Jour. on Rob. and Autom.*, 1987.

[22] Y. Hong, Q. Wu, Y. Qi, *et al.*, "A Recurrent Vision-and-Language BERT for Navigation," in *CVPR*, 2021.

[23] R. Loynd, R. Fernandez, A. Celikyilmaz, *et al.*, "Working memory graphs," in *ICML*, 2020.

[24] S. Ren, K. He, R. Girshick, *et al.*, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *TPAMI*, 2017.

[25] R. Krishna, Y. Zhu, O. Groth, *et al.*, "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations," *IJCV*, 2017.

[26] T. Gupta, A. Schwing, and D. Hoiem, "Vico: Word embeddings from visual co-occurrences," in *ICCV*, 2019.

[27] G. I. Magalhaes, V. Jain, A. Ku, *et al.*, "General evaluation for instruction conditioned navigation using dynamic time warping," in *NeurIPS*, 2019.

[28] S. Salvador and P. K. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, 2004.

[29] W. Hao, C. Li, X. Li, *et al.*, "Towards Learning a Generic Agent for Vision-and-Language Navigation via Pre-training," *CVPR*, 2020.

[30] V. Jain, G. Magalhaes, A. Ku, *et al.*, "Stay on the path: Instruction fidelity in vision-and-language navigation," in *ACL*, 2020.

[31] D. Fried, R. Hu, V. Cirik, *et al.*, "Speaker-follower models for vision-and-language navigation," in *NeurIPS*, 2018.

[32] C. Y. Ma, Z. Wu, G. Alregib, *et al.*, "The regretful agent: Heuristic-aided navigation through progress estimation," in *CVPR*, 2019.

[33] X. Wang, Q. Huang, A. Celikyilmaz, *et al.*, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *CVPR*, 2019.

[34] F. Landi, L. Baraldi, M. Cornia, *et al.*, "Multimodal attention networks for low-level vision-and-language navigation," in *CVIU*, 2021.

[35] Z. Deng, K. Narasimhan, and O. Russakovsky, "Evolving graphical planner: Contextual global planning for vision-and-language navigation," in *NeurIPS*, 2020.

[36] H. Tan, L. Yu, and M. Bansal, "Learning to navigate unseen environments: Back translation with environmental dropout," in *NAACL*, 2019.

[37] Y. Qi, Z. Pan, S. Zhang, *et al.*, "Object-and-Action Aware Model for Visual Language Navigation," in *LNCS*, 2020.

[38] W. Zhu, H. Hu, J. Chen, *et al.*, "BabyWalk: Going Farther in Vision-and-Language Navigation by Taking Baby Steps," in *ACL*, 2020.

[39] Y. Hong, C. Rodriguez-Opazo, Y. Qi, *et al.*, "Language and Visual Entity Relationship Graph for Agent Navigation," in *NeurIPS*, 2020.