TP-MDDN: Task-Preferenced Multi-Demand-Driven Navigation with Autonomous Decision-Making

Shanshan Li[1], Da Huang[23], Yu He[13], Yanwei Fu[13†], Yu-Gang Jiang[1], Xiangyang Xue[1†]

[1]Fudan University   [2]Shanghai Jiao Tong University

[3]Shanghai Innovation Institution

Abstract

In daily life, people often move through spaces to find objects that meet their needs, posing a key challenge in embodied AI. Traditional Demand-Driven Navigation (DDN) handles one need at a time but does not reflect the complexity of real-world tasks involving multiple needs and personal choices. To bridge this gap, we introduce **Task-Preferenced Multi-Demand-Driven Navigation (TP-MDDN)**, a new benchmark for long-horizon navigation involving multiple sub-demands with explicit task preferences. To solve TP-MDDN, we propose **AWMSystem**, an autonomous decision-making system composed of three key modules: Break-LLM (instruction decomposition), LocateLLM (goal selection), and StatusMLLM (task monitoring). For spatial memory, we design MASMap, which combines 3D point cloud accumulation with 2D semantic mapping for accurate and efficient environmental understanding. Our Dual-Tempo action generation framework integrates zero-shot planning with policy-based fine control, and is further supported by an Adaptive Error Corrector that handles failure cases in real time. Experiments demonstrate that our approach outperforms state-of-the-art baselines in both perception accuracy and navigation robustness.

1   Introduction

In daily life, people often identify a need and look for something in their environment to meet it [3, 4]. Demand-Driven Navigation (DDN) [1] is a task where an agent receives a natural language instruction (e.g., "I am tired") and must find an object that fulfills that need (e.g., a bed or chair). This is a variation of the ObjectNav task. However, a single need can often be met in different ways, depending on personal preferences. For example, "organize the living space" could mean finding cleaning tools, decorative items, or storage boxes. To guide the agent effectively, it is important to clarify the user's specific preference, like focusing on decoration, so the instruction becomes actionable. People also tend to have a series of needs, such as cleaning, then resting, then eating. Efficiently handling multiple needs and evaluating the success of these actions is still a major challenge. This paper explores how to enhance long-horizon DDN tasks by making user preferences more explicit.

Recent work like MO-DDN [5] addresses navigation instructions through Multi-object Demand-driven Navigation with human preferences. For example, given an instruction like "I need to display my photography collection, preferably with good lighting", the demand may involve multiple objects such as picture frames, bookshelves, and ceiling lamps. However, it still focuses on single-demand navigation. Some works have explored long-range instruction navigation by constructing Landmark Semantic Memory for decision-making planning [6] or adopting Autonomous Evolution mechanisms [7, 8], but they primarily target object-driven navigation.
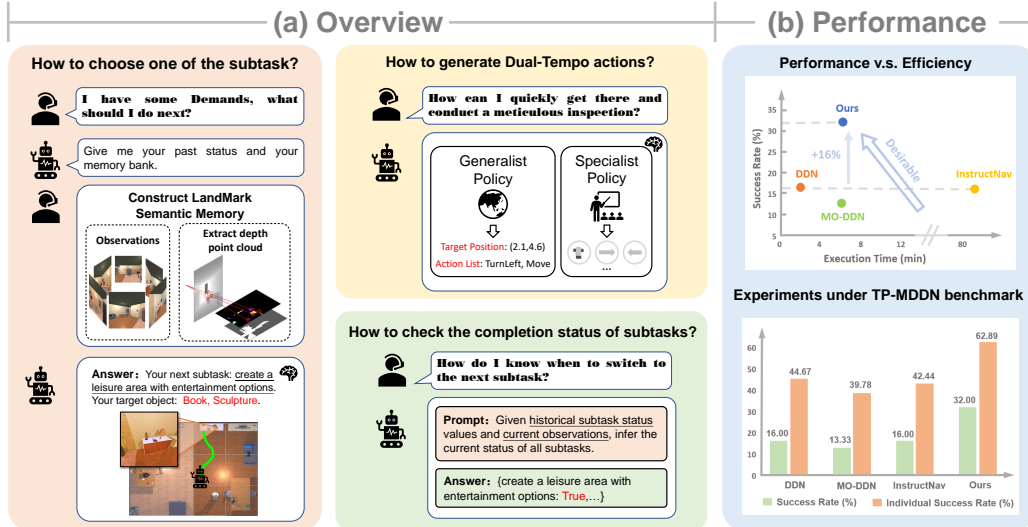

[†]Corresponding Authors

Figure 1: This figure presents our autonomous decision-making process and its performance benefits. (a) Overview: The pink area shows subtask selection using landmark semantic memory; the yellow area explains Dual-Tempo action generation via generalist and specialist policies; the green area details dynamic subtask completion checks. (b) Performance: Our method achieves a 16% higher success rate than DDN [1] and InstructNav [2] under the TP-MDDN benchmark, with superior efficiency and individual success rates, highlighting its effectiveness and reliability.

We introduce a new benchmark called Task-Preferenced Multi-Demand-Driven Navigation (TP-MDDN) to handle scenarios with multiple needs, where each need includes a clear preference for a specific task-related object category. In a simulated multi-room home environment, we use DeepSeekV3 [9] and GPT-4o [10] to generate long-horizon instructions and object pairs across different scenes. These pairs are manually checked to ensure accuracy.

To efficiently manage memory in long-term navigation, we introduce a novel Multidimensional Accumulated Semantic Map (MASMap) that achieves a balance between accuracy and efficiency without requiring additional training. MASMap integrates local 3D point cloud accumulation with a global 2D semantic map to build and maintain spatial memory over time. A core challenge of merging semantically similar objects viewed from different perspectives is addressed using IoU-based fusion and Ram-Grounded-SAM for accurate segmentation and labeling. To reduce storage overhead, we implement an efficient update-and-prune strategy that preserves critical small objects. Our global map structure further minimizes redundancy while ensuring consistent and reliable semantic memory across extended navigation trajectories.

To address long-horizon visual navigation tasks, we further propose the **Autonomous Decision-Making World Model System (AWMSystem)**, inspired by WMNav [11]. AWMSystem breaks down complex instructions into sub-demands using **BreakLLM**, dynamically selects goals through **LocateLLM** based on object memory and execution status, and tracks task progress via **StatusMLLM** using real-time observations. For lightweight deployment, we employ a dual-tempo action generation strategy: zero-shot planning using A* algorithm with obstacle maps and affordance estimation (as in InstructNav [2]), and fine-grained policy-based control near targets, following DDN [1]. A major challenge in simulation environments, i.e., handling logical loops, boundary violations, and unseen obstacles, is addressed with an **Adaptive Error Corrector**, which adjusts actions in real time based on environmental feedback, greatly enhancing robustness. This modular design supports efficient, reliable long-term navigation without requiring additional end-to-end retraining.

We summarize our main contributions as follows: (1) We introduce **TP-MDDN**, a new long-horizon navigation benchmark with multi-sub-demand tasks and explicit task preferences, featuring high semantic richness and scene diversity through the use of DeepSeekV3 and GPT-4o. (2) We propose **AWMSystem**, an autonomous decision-making world model system composed of BreakLLM, LocateLLM, and StatusMLLM, enabling efficient instruction decomposition, dynamic goal selection,

and real-time execution monitoring without requiring end-to-end training. (3) We design a lightweight **MASMap**, which fuses 3D object detection and 2D semantic mapping to achieve both accurate perception and computationally efficient navigation. (4) Extensive experiments demonstrate that our method **significantly outperforms state-of-the-art baselines**, with ablation studies confirming the effectiveness of each component. Our system achieves strong robustness and environmental adaptability while maintaining low computational overhead, showcasing a practical balance between performance and efficiency that has not been addressed by previous works.

2 Related Works

Vision-Language Navigation. Visual-Language Navigation (VLN) involves guiding an agent to a goal based on language instructions and visual observations. Early methods focused on progress estimation [12, 13], backtracking [14], reinforcement learning [15, 16], and policy learning [1, 5, 17, 18, 19, 20, 21]. Some works extracted object and action types from instructions [22, 23, 24, 25, 26], while others leveraged transformers for history encoding [27, 21, 22, 23, 28], built topological maps [29, 20, 30], or predicted future events [31]. Pre-training [32, 33, 34, 35, 36] and data enhancement [37, 38, 39, 35, 40, 41] have also been explored. However, these approaches struggle with long-horizon continuous navigation. In contrast, our method adapts zero-shot scene layout understanding to achieve strong performance in such challenging tasks.

Continuous VLN with Foundation Models. Foundation models, like LLMs, VLMs, and LVLMs, have advanced visual navigation [42, 43] by enabling strong reasoning, high-level planning, and end-to-end action generation abilities. Recent zero-shot approaches use these models for collaboration [44], memory-based reasoning [45, 46], or instruction tuning [47]. However, most work focuses on discrete actions, while continuous navigation, more suited for real-world use, remains difficult [48, 1]. Earlier methods used GRU or LSTM [49, 50], while recent ones address object-targeted [51, 52, 53, 54] and instruction-following tasks [55, 56, 57]. Some predict progress [55], use value maps [58], or plan trajectories [59, 60, 61]. Yet, long-horizon tasks suffer from the high cost of frequent LLM calls. To solve this, we introduce a Dual-Tempo action generator, inspired by dual-system robotics [62, 63, 5], which boosts efficiency without losing performance.

Long-Horizon Navigation. Long-horizon navigation is essential for building agents that can learn and act over time. While benchmarks like LH-VLN [64] have made progress, success rates remain low due to limited memory encoding. Recent work has explored memory-based methods—like WMNav [11] for relation prediction and Mem2Ego [6] for using landmarks to guide decisions—but they focus on object goal navigation. Real-world systems like ReMEmbR [65] support long-range navigation but lack continuous control. Minecraft agents show promise through skill reuse [8, 7], but they rely on explicit object prompts. In contrast, demand-driven navigation, where agents meet high-level needs like "find an office tool", better reflects real scenarios. Yet, handling multi-step goals without object-specific instructions remains difficult. To tackle this, we combine obstacle avoidance, error correction, status tracking, and large foundation models to boost navigation performance in complex environments.

3 Tasked-Preferred Multi-Demand-Driven Navigation

In a Task-Preferred Multi-Demand-Driven Navigation task, the agent is given a natural language long-horizon instruction (e.g., *"Organize the living space by arranging decorative items, set up a cozy entertainment corner with seating and media devices"*), comprising multiple subtasks. Each subtask combines a basic requirement and a task preference; for example, the basic requirement is *"set up a cozy entertainment corner"*, and the task preference is *"with seating and media devices"*.

Formally, let S denote the set of environments and D denote the set of long-horizon instructions. For a given instruction $d = \langle d_1, d_2, \ldots, d_L \rangle \in D$, where L is the number of subtasks and d_i is the i-th subtask, the goal is to determine whether the agent successfully finds objects satisfying each subtask. Specifically, in each episode: (1) A random environment $s \in S$ is selected. (2) The agent is initialized at a random position and orientation within s. (3) A random instruction d is chosen. (4) Let O be the set of objects in the environment. We define a function $G : D \times O \rightarrow \{0, 1\}^L$, mapping instructions and environment objects to a binary vector indicating whether each subtask is satisfied. To complete the instruction, the agent must find at least one object for each subtask. Each subtask is
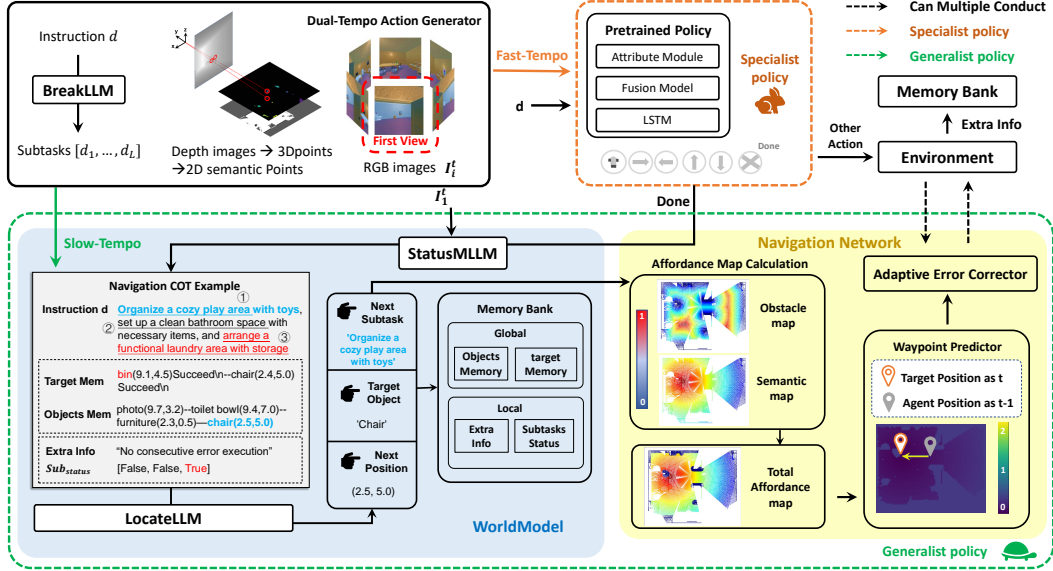
Figure 2: **Overview**. This diagram illustrates the dual-tempo action generation process in our system. The BreakLLM module decomposes the instruction. Then, depth images are converted into 2D semantic points. The fast-tempo branch uses a pretrained policy to generate primitive actions, while the slow-tempo branch employs LocateLLM for high-level navigation reasoning, determining target objects and positions. StatusMLLM tracks task progress and updates memory. The Navigation Network performs affordance map computation, adaptive error correction, and waypoint prediction.

strictly binary, defined as either success or failure. To reduce the difficulty of the TP-MDDN task, if the 2D Euclidean distance between the agent and the object is within a threshold ϵ_{dis}, the object is considered found. This differs from DDN [1], which requires finding the target object within the field of vision.

At each step, the agent can generate actions using either the **Generalist Policy** or the **Specialist Policy**. The Generalist Policy drives the agent toward a target pose selected by the large model, while the Specialist Policy uses a pretrained network to execute one of six actions: `MoveAhead`, `RotateRight`, `RotateLeft`, `LookUp`, `LookDown`, and `Done`. The system invokes the Generalist Policy at regular intervals to explore potentially relevant regions, after which the pretrained policy [1] is used to search for objects matching the task requirements. The agent terminates the navigation task when the number of steps reaches the maximum length Len_{max}, or when all subtasks are completed. From the test set of ProcTHOR [66], we generated 200 long-horizon instructions, each containing three subtasks, spanning 68 rooms.

3.1 Autonomous Decision-Making World Model System

AWMSystem Overview. Figure 2 illustrates our Autonomous Decision-Making World Model System, which models the environment and predicts actions based on observations. It constructs a real-time 2D semantic map with efficient memory storage and uses past trajectories and layout information to select the next target. The Dual-Tempo action generator plans actions, while an Adaptive Error Corrector adjusts strategies based on feedback.

3.1.1 Construction of Long-Range Memory Banks

Raw Data Processing. Building the memory bank consists of three components: input data processing, real-time accumulation, and recording of different types of historical data. For raw data processing, we perform object detection and segmentation on RGB images and extract 3D point clouds of detected objects from depth maps. These 3D points are then fused and compressed into a 2D semantic map. Specifically, at regular intervals, the agent performs environmental sensing. Suppose the current step is t; the agent captures n RGB images $I_1^t, \ldots, I_n^t$ and corresponding depth

images $\text{Depth}_1^t, \ldots, \text{Depth}_n^t$. For each image I_i^t, we use the Ram-Grounded-SAM model [67, 68] to obtain object labels, bounding boxes, and segmentation masks. Next, we compute the real-world 3D point cloud PC_{cur} from each depth image using the camera's intrinsic parameters and rotation matrix.

Real-time Accumulation. In the real-time accumulation design, since object point clouds obtained from different viewpoints at the current location may correspond to the same physical object, we design a point cloud update strategy. Let r be an object point cloud in the recorded set PC_R, r_{cur} be an object point cloud in PC_{cur} and let r_{cur}^{final} denote the residual point cloud obtained by removing regions of r_{cur} that overlap with any object in PC_R. During this process, for each candidate $r \in PC_R$, we compute two overlap metrics based on the intermediate point cloud r_{cur}^* at a given stage:

$$os^* = \frac{\text{overlap\_score}(r_{cur}^*, r)}{|r_{cur}^*.pcd|}, \quad ros^* = \frac{\text{overlap\_score}(r_{cur}^*, r)}{|r.pcd|},$$

where the overlap score is computed using element-wise Euclidean distance between point clouds and $|\cdot|$ denotes the number of points. Let $\text{Update}(r_{cur}, PC_R)$ denote the operation that updates the reference set PC_R based on r_{cur}. We define it as follows:

$$\text{Update}(r_{cur}, PC_R) = \begin{cases} PC_R \cup \{r_{cur}\} & \text{if } \max_{r \in PC_R} os^* < 0.25 \\ r^*.pcd \leftarrow \text{Merge}(r.pcd, r_{cur}^{final}.pcd) & \\ r^*.class \leftarrow r_{cur}.class & \text{if } os^* > 0.8 \wedge ros^* > 0.8 \\ PC_R[r] \leftarrow r^* & \end{cases}$$

If the maximum os^* over all $r \in PC_R$ is less than 0.25, r_{cur} has negligible overlap with any existing object, so it is treated as a new object and added to PC_R. If both $os^* > 0.8$ and $ros^* > 0.8$ for a specific r, this indicates strong overlap, suggesting that r_{cur} and r represent the same object. In this case, their point clouds are merged, and the class label of r is updated to that of r_{cur}.

Fusion of Global Semantic Map. After recording the center coordinates of object point clouds, all 3D point cloud data is cleared to save memory. Let $\mathcal{OM}_t$ denote the object memory bank storing information about previously detected objects up to time step t. Each object's 2D information is recorded as $\{\texttt{class}: c_{obj} \in \mathcal{C}, \texttt{center}: \mathbf{p_{obj}} = (x_{obj}^{center}, y_{obj}^{center}), \texttt{bbox}: [x_{obj}^{min}, x_{obj}^{max}, y_{obj}^{min}, y_{obj}^{max}]\}$, where $\mathcal{C}$ is the set of possible object classes. We compute the 2D IoU between current and historical objects and apply the Hungarian algorithm to find the most similar historical object. If a match exists, we update the corresponding entry in the 2D semantic map and object memory bank; otherwise, we add the new object. Under a task, we continuously accumulate the names and locations of explored objects to form a target memory. The data in the target memory bank is formatted as $\langle$Target Object, (x, y), Feedback Type$\rangle$, where (x, y) denotes the target location on the map, and *Feedback Type* includes: `Success`, `Obstructed`, `Out-of-Bounds`, and other failure descriptions.

Explanation of Memory Bank. As illustrated in Figure 2, the memory bank contains two components. Global Records serve as Long-Term Memory by storing visited information, including a cumulative map of detected objects with their 2D poses and a history of planned targets with execution outcomes, enabling continuous progress tracking. In contrast, Local Updates function as Short-Term Memory by maintaining transient data for immediate decision-making. This includes local 3D point clouds extracted from the current panoramic view, additional information derived from past execution failures to address recurring issues, and the current status of the ongoing subtask.

3.1.2 Summary of Foundation Model Usage

BreakLLM. We employ a Large Language Model (LLM) to automatically decompose long-horizon instructions into a subtask list d_{sub}, and initialize a corresponding subtask execution status list Sub_{Status}, which is maintained throughout the task. The instruction decomposition is formalized as $(d_{sub}, Sub_{Status}) = BreakLLM(d)$, where all entries in Sub_{Status} are initially set to `False`.

LocateLLM. At time step t, we maintain a memory of detected objects and their 2D coordinates, denoted as $\mathcal{OM}_t$ (Object Memory). The primary inputs to the decision-making module include: the overall instruction d, the subtask list d_{sub}, the current subtask status Sub_{Status}, the target memory $\mathcal{T}_t$, and the object memory $\mathcal{OM}_t$. Large models may struggle to fully comprehend long sequences of historical trajectories, potentially leading to repeated failures on the same object. To mitigate this, we introduce auxiliary feedback to help the model detect and avoid execution
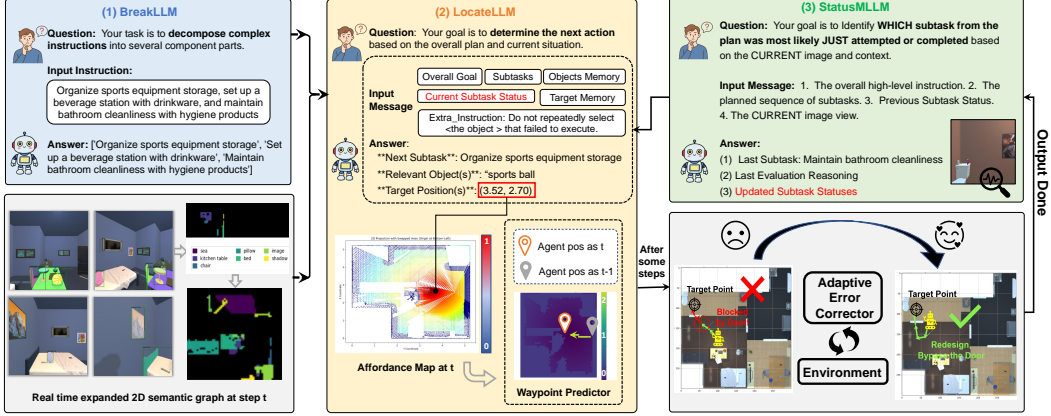
Figure 3: **Foundation Model Usage.** BreakLLM decomposes the instruction. The agent uses Ram-Grounded-Sam [67, 68] to segment panoramic RGB-D images and ultimately map them onto 2D semantic maps to form object memory. LocateLLM receives multiple types of data and outputs the next target object and position. StatusMLLM determines whether a subtask has been completed based on the current observed image. Adaptive error corrector re-plans the failed trajectory.

loops. Specifically, we track the number of consecutive failed attempts n_{CFE} on the target object $TargetObj_{t-1}$. If $n_{\text{CFE}} \geqslant n_{\text{tolerance}}$, we generate an auxiliary prompt $Extra_{\text{info}}$ in the form: "Do not select `<object>` or `<position>` in the next step". This prompt is passed to the planner to discourage revisiting failed targets. The next target object is determined by:

$$TargetObj_t = \text{LocateLLM}\left(\mathcal{T}_t, \mathcal{OM}_t, Sub_{Status}, d, d_{sub}, \mathbb{I}\left(n_{\text{CFE}} \geqslant n_{\text{tolerance}}\right) \cdot Extra_{\text{info}}\right) \quad (1)$$

StatusMLLM. Iterative updating of subtask completion status is crucial for long-horizon task navigation. We introduce a multimodal LLM, StatusMLLM, to update Sub_{Status} when the policy network outputs a Done action. This design is motivated by the observation that prior methods often trigger Done upon detecting an object that matches the instruction [21, 1]. We leverage this behavior to infer subtask completion. Specifically, we input the overall instruction d, the previous subtask status Sub_{Status}^{t-1}, the target memory $\mathcal{T}_t$, and the current image I_1^t into StatusMLLM. The model outputs inference on which subtask might have just been completed Sub_{cur}, explanation for its judgment $Reason$, updated subtask completion status list Sub_{status}^t. This process is formalized as:

$$\left(Sub_{cur}, Reason, Sub_{Status}^t\right) = \text{StatusMLLM}\left(d, Sub_{Status}^{t-1}, \mathcal{T}_t, I_1^t\right) \cdot \mathbb{I}\left(A_{\text{policy}} = \text{Done}\right) \quad (2)$$

3.1.3 Dual-Tempo Action Generator

Inspired by prior work on two-stage navigation strategies [52, 11, 62, 63, 5] and considering that frequent invocation of large models at every step incurs significant computational overhead, we propose the **Dual-Tempo Action Generator**, as illustrated in Figure 2. This architecture decouples planning into a slow-tempo phase and a fast-tempo phase to balance reasoning depth with efficiency.

In the slow-tempo execution phase, we follow these steps: (1) Extract the current object point cloud PC_{cur} from the panoramic observation and fuse it into the object memory $\mathcal{OM}_t$. (2) Feed historical context and execution feedback into LocateLLM to determine the next target object and its 2D location. (3) Compute affordance value maps that encodes navigational feasibility and semantic relevance. (4) Apply the A* algorithm on the affordance map to generate a globally feasible navigation path. (5) Decompose the path into a sequence of intermediate waypoints, convert each segment into discrete actions, and execute them sequentially. (6) Invoke the **Adaptive Error Corrector** to detect and rectify trajectory failures.

In the fast-tempo execution phase, we directly employ the pretrained policy from prior work [1], which outputs low-level actions based on the current RGB image and the high-level instruction. When the policy outputs Done, the StatusMLLM module is triggered to evaluate whether a subtask

has been completed. Next, we detail the computation of the affordance value map and the operational mechanism of the Adaptive Error Corrector.

Calculation of Affordance Map. At step t, we sequentially read an RGB image I_i^t and a depth image Depth_i^t from the panoramic views. For each depth image, we obtain its real-world 3D point cloud PC_{cur}. Based on the height of the points, we classify them into navigable points $\mathcal{N}_{\text{navi}}^{t,i}$ and obstacle points $\mathcal{O}^{t,i}$. Then, we project $\mathcal{N}_{\text{navi}}^{t,i}$ and $\mathcal{O}^{t,i}$ onto a 2D grid map to form $\mathcal{N}_{\text{grid}}^{t,i}$ and $\mathcal{O}_{\text{grid}}^{t,i}$, and perform dilation on the obstacle regions $\mathcal{O}_{\text{grid}}^{t,i}$.

Next, we construct the 2D affordance map by computing the obstacle avoidance affordance and semantic affordance. Following similar practices [2], we calculate the Euclidean distances from navigable points $\mathcal{N}_{\text{navi}}^{t,i}$ to category-specific point sets. For any navigable point $n_i \in \mathcal{N}_{\text{navi}}^{t,i}$, if its distance to the 2D obstacle point cloud satisfies $d_{\text{obs}}(n_i) < \tau_{\text{obs}}$, the obstacle avoidance affordance value $a_{\text{obs}}(n_i)$ is set to 0; otherwise, it is normalized as $a_{\text{obs}}(n_i) = (d_{\text{obs}}(n_i) - d_{\text{min}})/(d_{\text{max}} - d_{\text{min}})$. From the target 2D coordinates obtained in Section 3.1.2, we define the semantic affordance value $a_{\text{tgt}}(n_i)$ as the inverse of the normalized distance to the target point cloud: $a_{\text{tgt}}(n_i) = 1 - (d_{\text{tgt}}(n_i) - d'_{\text{min}})/(d'_{\text{max}} - d'_{\text{min}})$, where $d_{\text{tgt}}(n_i)$ represents the distance from n_i to the target point. This means that positions closer to the target receive higher semantic affordance values. Here, d_{min} and d_{max} denote the minimum and maximum distances to the obstacle point cloud, respectively; d'_{min} and d'_{max} are the corresponding minimum and maximum distances to the target point cloud. Finally, we compute the final affordance value $a_{\text{final}}(n_i)$: if $a_{\text{obs}}(n_i) = 0$, then $a_{\text{final}}(n_i)$ is set to 0; otherwise, it is set to the clipped value of $a_{\text{tgt}}(n_i)$ between 0.1 and 1. The formula is as follows:

$$a_{\text{final}}(n_i) = \begin{cases} 0, & a_{\text{obs}}(n_i) = 0 \\ \text{clip}\left(a_{\text{tgt}}(n_i),\ 0.1,\ 1\right), & \text{otherwise} \end{cases} \tag{3}$$

Adaptive Error Corrector. Our Adaptive Error Corrector uses environmental feedback to correct navigation errors. When the agent detects that a `MoveAhead` action may lead to a collision (e.g., with doors, walls, or furniture), it re-plans a new path from the current position and updates the waypoint sampling strategy. Under normal operation, the agent navigates by sampling a waypoint every n_{waypoint} steps, using a discrete action space defined by forward translations of 0.25 meters and rotational increments of 30 degrees. When re-planning is triggered due to execution failure, the agent continues to operate within the same action space. The affordance map is recomputed based on the current state, allowing the planner to generate a revised trajectory that avoids obstacles and resumes progress toward the target object. To improve navigation precision, the trajectory is divided into two segments: an initial segment and a subsequent segment. In the initial segment, a finer sampling interval n_{block} is applied to support detailed spatial reasoning near obstacles. In the subsequent segment, the sampling frequency reverts to the standard rate n_{waypoint}, consistent with the regular navigation strategy.

4 Experiments

4.1 Experimental Setups

Experiments. We use AI2-THOR [69] as our simulator and ProcThor as our scene dataset [66]. We used each of DeepSeek-V3 [9] and GPT-4o [10] to generate 100 task-preferenced, multi-demand-driven unseen instructions in test scenarios (totaling 200 commands). In all experimental settings, the success distance threshold ϵ_{dis} is 1.5 meters, the maximum step count Len_{max} is 50, and the tolerance for repeated failed attempts on the same object $n_{\text{tolerance}}$ within $Extra_{\text{info}}$ is 2. The obstacle avoidance distance τ_{obs} used during affordance map computation is 0.25 meters. When processing input data, we set the camera resolution to 300×300 and the horizontal field of view (HFoV) to $90°$ for the agent. The agent operates in a closed-loop fashion, perceiving environmental feedback (e.g., success, collision, or boundary detection) immediately after executing each discrete action. All experiments can be run on a single NVIDIA H100 80GB GPU.

Evaluation Metrics. In line with prior works [1, 5, 64], we adopt the following evaluation metrics: (1) Success Rate (SR): The proportion of tasks in which the agent successfully reaches the target object associated with each subtask. (2) Independent Success weighted by Path Length (ISPL): For each task, the success of each subtask is weighted by the ratio of the shortest path length to the
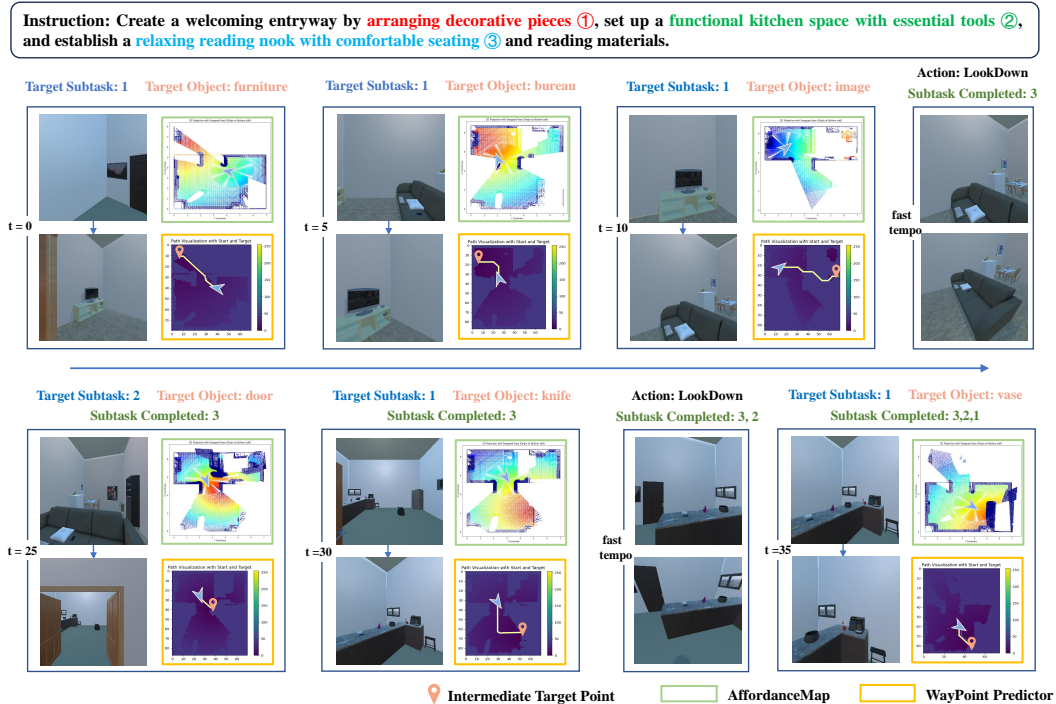
Figure 4: **Visualization Results**. The intelligent agent receives a Task-Preferenced Multi-Demand-Driven instruction, autonomously decomposes it into multiple subtasks, and identifies objects in the scene that match the unexecuted subtasks. On the affordance maps, redder values indicate higher affordance scores. The arrow in the waypoint predictor graph represents the agent's location and current field of view. As the step count increases, the three subtasks are gradually completed.

actual path length, then averaged across all subtasks. (3) Successful Trajectory Length (STL): The average number of steps taken in successful navigation trials. (4) Independent Success Rate (ISR): The success rate for each subtask evaluated individually. We evaluate the above metrics over 50 tasks for each method, averaging three runs due to the randomness in large model outputs.

Baselines. DDN is an end-to-end, single-demand-driven navigation method [1]. MO-DDN is an attribute-based exploration modular agent designed for multi-object, demand-driven navigation [5]. InstructNav uses dynamic chain-of-navigation and multi-sourced value maps to generate robot-actionable trajectories [2]. These three methods are the most relevant and state-of-the-art in the field of demand-driven navigation.

4.2 Main Results

The experimental results with baselines are shown in Table 1. Our AWM-Nav achieves the highest scores across all metrics, demonstrating its superior performance in long-horizon navigation tasks. In contrast, all baseline methods achieve significantly lower results than ours, which can be attributed to the fact that long-horizon instruction navigation typically requires stronger active exploration capabilities and the ability to execute multiple subtasks, while these baselines are designed for single-task navigation and lack such capabilities. InstructNav [2] combines zero-shot reasoning using large models with explicit memory mechanisms. However, its planned paths often result in collisions within the environment. Although InstructNav employs dynamic chain-of-thought reasoning over actions, the LLM struggles to infer the completion status of subtasks from raw action sequences due to limited contextual understanding. MO-DDN [5] adopts a two-stage navigation process consisting of coarse search followed by fine-grained localization. As the full implementation of MO-DDN has not been publicly released, we instead integrate its policy network with our own MASMap for the coarse exploration stage, achieving the lowest success rate. Despite showing some capability in multimodal alignment, MO-DDN still falls short when coping with the challenges posed by our

Table 1: Comparison with state-of-the-art methods on the TP-MDDN benchmark. In the Large-model Inference column, ✓ indicates the LLM is used for reasoning. "Explicit History" refers to methods that record object names and positions in the scene.

Method	Zero-shot	Large-model Inference	Explicit History	STL↑	ISR↑	SR↑	ISPL↑
DDN [1]	✗	✗	✗	15.50	44.67	16.00	40.66
MO-DDN [5]	✗	✓	✓	12.11	39.78	13.33	36.25
InstructNav [2]	✓	✓	✓	9.50	42.44	16.00	39.41
AWM-Nav	✗	✓	✓	**20.11**	**62.89**	**32.00**	**44.19**

Table 2: Ablation results for object segmenters, reasoning large models, Adaptive Error Corrector, and StatusMLLM. **-** means no information in this line, ✓ means using the method, and ✗ means not using the method.

1. The effect of different object segmenters					2. Different reasoning large models				
Method	STL↑	ISR↑	SR↑	ISPL↑	Method	STL↑	ISR↑	SR↑	ISPL↑
GLEE [70]	14.94	51.11	21.33	41.05	Qwen2.5-VL-7B	10.97	47.78	19.33	36.45
YOLO	15.56	58.00	29.33	43.69	GPT-4o	17.51	56.44	28.67	39.95
RAM-Grounded-SAM [67, 68]	**20.11**	**62.89**	**32.00**	**44.19**	**Qwen2-5-VL-72B**	**20.11**	**62.89**	**32.00**	**44.19**

3. Influence of Adaptive Error Corrector						4. The effect of StatusMLLM				
BlockCorret	BeyondCorret	STL↑	ISR↑	SR↑	ISPL↑	With/Without	STL↑	ISR↑	SR↑	ISPL↑
✗	✓	13.49	59.33	27.33	43.25	-	-	-	-	-
✓	✗	16.86	60.44	28.00	42.20	✗	15.94	60.67	27.33	42.46
✓	✓	**20.11**	**62.89**	**32.00**	**44.19**	✓	**20.11**	**62.89**	**32.00**	**44.19**

task-preferenced, multi-demand navigation scenarios. Note that STL refers to the average length of successful trajectories, and our higher score is due to solving some long-distance tasks that require crossing rooms, which increases the average trajectory length.

Besides the metrics in this table, we also pay particular attention to the average execution time, shown in Figure 1. In continuous-action navigation, using LLM inference at every step can be very time-consuming, as demonstrated by InstructNav's performance in (b). The average execution time per long-horizon instruction is 6.82 minutes for AWM-Nav, 1.74 minutes for DDN [1], 6.79 minutes for MO-DDN [5], and 88.90 minutes for InstructNav [2]. In the detailed time breakdown, slow-paced actions account for approximately 22 times the duration allocated to fast-paced actions in our method. In summary, our method adopts a dual-tempo action generator to save time and uses an automatic decision-making system built with large models to enhance the agent's reasoning capability.

Ablation Studies. As shown in Table 2, regarding the effect of **different object segmenters**, the RAM-Grounded-SAM-based [67, 68] model achieves the best performance, while GLEE [70] lacks precision and YOLO (Ultralytics YOLOv11) performs suboptimally. For different **reasoning large models**, the well-known GPT-4o does not lead to significant improvements, possibly due to the strong context understanding and state switching awareness required in long-horizon navigation tasks. The open-source Qwen2-5-VL-72B [71, 72] achieves the best metrics. After examining the execution behavior of the agent, it was found that the number of parameters in the large model affects the performance of intelligent planning. In evaluating the influence of the **Adaptive Error Corrector**, replanning affordance maps proves effective, and unexpected situations remain important to monitor and avoid, even with strong large model reasoning capabilities. With regard to **the effect of StatusMLLM**, task status tracking is crucial for long-horizon instructions, as misjudgment or absence of status reasoning can severely impact the success of the entire trajectory.

5   Conclusion and Discussion

This paper introduces a new benchmark, TP-MDDN, to address navigation tasks involving multiple sub-demands and explicit task preferences. Meanwhile, it proposes the AWMSystem decision-making system, MASMap spatial memory scheme, Dual-Tempo action generation framework, and an adaptive error corrector, which effectively tackle the challenges in TP-MDDN. Experiments show that the method achieves higher navigation accuracy than existing baselines and offers faster inference speed. However, the method has issues such as involuntary mode switching in the dual-tempo action generation framework and navigation decision errors caused by instruction misjudgment due to over-reliance on pre-trained large language models. Future work includes optimizing the mode switching

of the action generation framework through reinforcement learning and training domain-specific language models to reduce dependence on pre-trained models.

6 Acknowledgments

References

[1] Hongcheng Wang, Andy Guan Hong Chen, Xiaoqi Li, et al. Find what you want: Learning demand-conditioned object attribute space for demand-driven navigation. *Advances in Neural Information Processing Systems*, 36:16353–16366, 2023.

[2] Yuxing Long, Wenzhe Cai, Hongcheng Wang, et al. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024.

[3] Robert J Taormina and Jennifer H Gao. Maslow and the motivation hierarchy: Measuring satisfaction of the needs. *The American journal of psychology*, 126(2):155–177, 2013.

[4] Abraham Harold Maslow. *Motivation And Personality: Motivation And Personality: Unlocking Your Inner Drive and Understanding Human Behavior by AH Maslow*. Prabhat Prakashan, 2023.

[5] Hongcheng Wang, Peiqi Liu, Wenzhe Cai, et al. Mo-ddn: A coarse-to-fine attribute-based exploration agent for multi-object demand-driven navigation. *Advances in Neural Information Processing Systems*, 37:64176–64214, 2024.

[6] Lingfeng Zhang, Yuecheng Liu, Zhanguang Zhang, et al. Mem2ego: Empowering vision-language models with global-to-ego memory for long-horizon embodied navigation. *arXiv preprint arXiv:2502.14254*, 2025.

[7] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, et al. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[8] Tongtong Feng, Xin Wang, Zekai Zhou, et al. Evoagent: Agent autonomous evolution with continual world model for long-horizon tasks. *arXiv preprint arXiv:2502.05907*, 2025.

[9] DeepSeek-AI. Deepseek-v3 technical report, 2024.

[10] Aaron Hurst, Adam Lerer, Adam P Goucher, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[11] Dujun Nie, Xianda Guo, Yiqun Duan, et al. Wmnav: Integrating vision-language models into world models for object goal navigation. *arXiv preprint arXiv:2503.02247*, 2025.

[12] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, et al. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*, 2019.

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

[14] Liyiming Ke, Xiujun Li, Yonatan Bisk, et al. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6741–6749, 2019.

[15] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, et al. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6629–6638, 2019.

[16] Xin Wang, Wenhan Xiong, Hongmin Wang, et al. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–53, 2018.

[17] Peter Anderson, Qi Wu, Damien Teney, et al. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[18] Daniel Fried, Ronghang Hu, Volkan Cirik, et al. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[19] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of NAACL-HLT*, pages 2610–2621, 2019.

[20] Dong An, Yuankai Qi, Yangguang Li, et al. Bevbert: Multimodal map pre-training for language-guided navigation. *arXiv preprint arXiv:2212.04385*, 2022.

[21] Zihan Wang, Xiangyang Li, Jiahao Yang, et al. Gridmm: Grid memory map for vision-and-language navigation. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 15625–15636, 2023.

[22] Yuankai Qi, Zizheng Pan, Yicong Hong, et al. The road to know-where: An object-and-room informed sequential bert for indoor vision-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1655–1664, 2021.

[23] Abhinav Moudgil, Arjun Majumdar, Harsh Agrawal, et al. Soat: A scene-and object-aware transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 34:7357–7367, 2021.

[24] Yuankai Qi, Zizheng Pan, Shengping Zhang, et al. Object-and-action aware model for visual language navigation. In *European conference on computer vision*, pages 303–317. Springer, 2020.

[25] Chen Gao, Jinyu Chen, Si Liu, et al. Room-and-object aware knowledge reasoning for remote embodied referring expression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3064–3073, 2021.

[26] Xiangru Lin, Guanbin Li, and Yizhou Yu. Scene-intuitive agent for remote embodied visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2021.

[27] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, et al. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847, 2021.

[28] Yicong Hong, Qi Wu, Yuankai Qi, et al. Vln bert: A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1643–1653, 2021.

[29] Dong An, Hanqing Wang, Wenguan Wang, et al. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[30] Hao-Tien Lewis Chiang, Zhuo Xu, Zipeng Fu, et al. Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs. *arXiv preprint arXiv:2407.07775*, 2024.

[31] Zihan Wang, Xiangyang Li, Jiahao Yang, et al. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13753–13762, 2024.

[32] Weituo Hao, Chunyuan Li, Xiujun Li, et al. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13137–13146, 2020.

[33] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, et al. Improving vision-and-language navigation with image-text pairs from the web. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 259–274. Springer, 2020.

[34] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, et al. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16537–16547, 2022.

[35] Zun Wang, Jialu Li, Yicong Hong, et al. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12009–12020, 2023.

[36] Bowen Pan, Rameswar Panda, SouYoung Jin, et al. Langnav: Language as a perceptual representation for navigation. *arXiv preprint*, arXiv:2310.07889, 2023.

[37] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, et al. Airbert: In-domain pretraining for vision-and-language navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1634–1643, 2021.

[38] Kunyang Lin, Peihao Chen, Diwei Huang, et al. Learning vision-and-language navigation from youtube videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8317–8326, 2023.

[39] Jialu Li, Hao Tan, and Mohit Bansal. Envedit: Environment editing for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15407–15417, 2022.

[40] Keji He, Chenyang Si, Zhihe Lu, et al. Frequency-enhanced data augmentation for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 36:4351–4364, 2023.

[41] Jialu Li and Mohit Bansal. Panogen: Text-conditioned panoramic environment generation for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 36:21878–21894, 2023.

[42] Jiazhao Zhang, Kunyu Wang, Shaoan Wang, et al. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024.

[43] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, et al. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR, 2023.

[44] Yuxing Long, Xiaoqi Li, Wenzhe Cai, et al. Discuss before moving: Visual language navigation via multi-expert discussions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17380–17387. IEEE, 2024.

[45] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7641–7649, 2024.

[46] Jiaqi Chen, Bingqian Lin, Ran Xu, et al. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[47] Gengze Zhou, Yicong Hong, Zun Wang, et al. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. In *European Conference on Computer Vision*, pages 260–278. Springer, 2024.

[48] Jacob Krantz, Erik Wijmans, Arjun Majumdar, et al. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer, 2020.

[49] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[50] Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[51] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, et al. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation*, pages 42–48. IEEE, 2024.

[52] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, et al. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023.

[53] Yuxuan Kuang, Hai Lin, and Meng Jiang. Openfmnav: Towards open-set zero-shot object navigation via vision-language foundation models. *arXiv preprint arXiv:2402.10670*, 2024.

[54] Hang Yin, Xiuwei Xu, Zhenyu Wu, et al. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *Advances in Neural Information Processing Systems*, 37:5285–5307, 2024.

[55] Yanyuan Qiao, Wenqi Lyu, Hui Wang, et al. Open-nav: Exploring zero-shot vision-and-language navigation in continuous environment with open-source llms. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6710–6717. IEEE, 2025.

[56] Peihao Chen, Xinyu Sun, Hongyan Zhi, et al. A^{2}nav action-aware zero-shot robot navigation by exploiting vision-and-language ability of foundation models. *arXiv preprint arXiv:2308.07997*, 2023.

[57] Zhiyuan Li, Yanfeng Lu, Yao Mu, et al. Cog-ga: A large language models-based generative agent for vision-language navigation in continuous environments. *arXiv preprint arXiv:2409.02522*, 2024.

[58] Kehan Chen, Dong An, Yan Huang, et al. Constraint-aware zero-shot vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[59] Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, Mohamed Elnoor, et al. Convoi: Context-aware navigation using vision language models in outdoor and indoor environments. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13837–13844. IEEE, 2024.

[60] Soroush Nasiriany, Fei Xia, Wenhao Yu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

[61] Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. End-to-end navigation with vision language models: Transforming spatial reasoning into question-answering. *arXiv preprint arXiv:2411.05755*, 2024.

[62] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[63] Qingwen Bu, Hongyang Li, Li Chen, et al. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*, 2024.

[64] Xinshuai Song, Weixing Chen, Yang Liu, et al. Towards long-horizon vision-language navigation: Platform, benchmark and method. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12078–12088, 2025.

[65] Abrar Anwar, John Welsh, Joydeep Biswas, et al. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2838–2845. IEEE, 2025.

[66] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, et al. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022.

[67] Tianhe Ren, Shilong Liu, Ailing Zeng, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[68] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024.

[69] Eric Kolve, Roozbeh Mottaghi, Winson Han, et al. Ai2-thor: An interactive 3d environment for visual ai. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[70] Junfeng Wu, Yi Jiang, Qihao Liu, et al. General object foundation model for images and videos at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3783–3795, 2024.

[71] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[72] An Yang, Baosong Yang, Binyuan Hui, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction explain the task domain of Task Preferred Multi Demand Driven Navigation, and also clarify the contribution of proposing an autonomous decision-making system.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 discusses the limitations of involuntary mode switching in the dual-tempo action generation framework.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: There is no theoretical proof in this article, it is an innovation in other aspects.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: This article describes the experimental setup, dataset, and evaluation metrics in the fourth section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code declaration will be open source.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The model to be used, the specific method, and how to test it have all been explained clearly in the article.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: My experiment set the number of seeds and did not involve significance analysis.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In the fourth section, it was explained that all experiments can be run on a single NVIDIA H100 80GB.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: This study involves algorithm development and evaluation based on standard optimization benchmarks. It does not involve human subjects or applications that are clearly ethically sensitive, and we assume it complies with NeurIPS ethical standards.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: This article focuses on technological contributions and does not specifically discuss broader positive or negative social impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This article proposes a new optimization algorithm/framework, and the model and standard benchmark data used have not been found to have a high risk of abuse

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer:[Yes]

Justification: This article correctly cites the sources of existing assets, such as baseline methods and data sources.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The main new asset is the source code of the proposed method, which will be open-source.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer:[NA]

Justification: This study does not involve crowdsourcing experiments or research with humans as subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The study does not involve human subjects, therefore IRB approval is not applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The usage of the large model has been clearly explained in the third section.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.