Memo: An Open-World Agent with Multi-Scale Exploration and Memory-Driven Navigation in Minecraft

Anonymous CVPR submission

Paper ID 25998

Abstract

Autonomous completion of sparse, long-horizon tasks in open-world environments remains a key challenge for embodied intelligence. Minecraft, with its vast maps, dynamic entities, and complex task chains, provides an ideal benchmark. Existing hierarchical agents often suffer from inefficient exploration and limited semantic memory, causing repeated failures. To address this, we propose Memo (Multi-Scale Exploration and Memory-Driven Navigation), an embodied agent integrating multi-scale exploration with hierarchical memory-driven planning. Memo's Multi-Scale Progressive Exploration efficiently covers global, regional, and local areas, while its Dynamic Planner with Long–Short-Term Memory constructs a semantic knowledge graph to enable context-aware memory retrieval and adaptive task prioritization. Experiments show Memo outperforms prior methods, increasing exploration coverage by 14%, improving sequential task success by 30% with a 36% reduction in completion time, and boosting non-sequential task success by 44% with a 42% shorter execution time. These results demonstrate Memo's ability to combine efficient exploration, memory-driven reasoning, and adaptive planning for robust open-world task execution. We plan to release our code to support further research.

1. Introduction

With the rapid advancement of large-scale foundation models, embodied intelligence is shifting from single-task capabilities toward general-purpose agents that operate robustly across diverse, unstructured environments [4, 14, 15, 19]. A key requirement for such generality is the ability to autonomously solve sparse, multi-step instruction tasks in open-world settings. These tasks demand not only locating distant or rare targets across large areas but also adapting navigation and decision-making policies in response to evolving environmental conditions—posing a fundamental challenge for general embodied agents.

Minecraft, as a representative open-world sandbox environment, naturally exhibits these complexities. Its vast and personalized maps, dynamic entities, unknown resources, and long task chains—from resource gathering and crafting to construction and combat—closely mirror real-world challenges. Consequently, Minecraft has emerged as a standard testbed for evaluating the capability of embodied agents to perform long-horizon, sequential tasks [2]. Within such environments, the controller, responsible for exploration, navigation, and grounding high-level instructions, is critical to overall agent performance.

Most existing agent architectures adopt a hierarchical framework, where high-level planners decompose instructions into subtask sequences [16, 24], while controllers execute actions, explore the environment, and manage memory [17]. Despite their promise, current agents suffer from two major limitations. First they often exhibit limited semantic understanding and inflexible planning. Their memory system operate at a data level, lacking mechanisms to construct or utilize structures from historical experience. This prevents the effective identify and reuse of sparse but informative past observations, leading agents to repeatedly fail in similar situations. Moreover, their fixed decision pipelines struggle to adapt when target objects vary or when unexpected events arise, often resulting in suboptimal or failed trajectories. Second, existing exploration strategies tend to be locally greedy and globally inefficient, leaving large regions unvisited while generating redundant trajectories in already explored areas. Such inefficiency becomes especially problematic in large-scale open worlds, where balanced and comprehensive exploration is vital for task completion.

To address these bottlenecks, we introduce **Memo**, a new open-world agent with **M**ulti-Scale **E**xploration and **M**emory-Driven Navigati**o**n. Memo is designed to robustly handle instruction-following tasks involving sparse, distant, and dynamically evolving targets in large-scale Minecraft environments. Its architecture comprises three core components: explorer, memory, and planner. The Multi-Scale Progressive Exploration framework operates across global,

CVPR #25998

regional, and local scales, enabling comprehensive environment coverage while optimizing exploration paths. Dynamic Planning with Hierarchical Memory Update combines long- and short-term memory to support semantic cognition. Unlike traditional data-level memory, this system constructs a structured semantic knowledge graph from real-time interactions, facilitating context-aware information retrieval. Retrieved information feeds directly into the planner, which dynamically prioritizes tasks based on environmental observations and memory cues, allowing flexible and efficient task execution that adapts to dynamic changes. Experimental results demonstrate Memo's effectiveness: compared to the Mrsteve(state of the art)[17] in large-scale exploration, coverage increased by 14% with notable efficiency gains; for sparse sequential tasks, success rates improved by 30% with a 35.8% reduction in completion time; and in sparse non-sequential tasks, success rates rose by 44% with a 42.5% shorter completion time.

Our main contributions are summarized as follows:

1. We present Memo, a new open-world embodied agent equipped with multi-scale exploration and memory-driven navigation. Memo effectively handles instruction-following tasks with sparse and distant targets in large-scale Minecraft environments.
2. We propose a multi-scale progressive exploration strategy that gathers complementary information at global, regional, and local levels, enabling efficient exploration and large-area coverage in open-world settings.
3. We develop a dynamic planning mechanism that continuously updates geometric and semantic memory to support adaptive navigation, particularly in tasks where target objects are diverse or can change during execution.

2. Related work

2.1. Low-Level Control in Minecraft

Early rule-based systems, such as CraftAssist [8], relied on predefined scripts for game operations and dialogue, lacking learning ability and adaptability. MineRL [9] introduced reinforcement learning with large-scale human demonstrations, enabling skill learning but struggling with long-horizon tasks. The data-driven shift began with VPT [1], which mapped vision to actions through large-scale video pretraining, enabling autonomous complex behaviors. Building on this, Steve-1 [13] employed a UNCLIP-based text–vision–action mapping, supporting open-text instruction following but without long-term memory for multi-scenario tasks. Memory-augmented approaches like MrSteve [17] incorporated a Place–Event Memory mechanism to link spatial, temporal, and event information, enhancing long-sequence execution. However, semantic understanding and memory management remain shallow, and exploration strategies still lack global planning and flexibility.

2.2. Exploration and Navigation Strategies

Efficient exploration is critical for open-world agents. Frequency-based methods [3, 23, 27] are simple but suffer from local redundancy and scalability issues. Video-pretrained navigation (e.g., VPT-Nav) adapts to dynamic environments but relies on local observations, leading to suboptimal trajectories in unseen scenes. Active information–gain strategies [4, 5, 29] discover novel areas effectively but are computationally intensive, violating Minecraft's real-time ($\leq$20 ms/step) constraints. Coverage-based approaches ensure completeness but struggle to balance coverage with task priorities and 3D spatial reasoning.

2.3. Memory Systems for Open-World Agents

Memory is essential for long-horizon reasoning and planning [10, 20–22, 28]. Existing systems often use single-modal or loosely fused multi-modal storage: Voyager [25] stores skills as text; GITM [31] integrates textual memory for efficient reasoning; MP5 [18] and JARVIS-1 [26] maintain multi-modal context for situational retrieval; Optimus-1 [12] enhances storage with a multi-modal experience pool. MrSteve [17] improves memory via a Place–Event mechanism, yet retrieval accuracy, semantic abstraction, and generalization remain limited. Overall, current memory systems lack robust semantic adaptability and dynamic updating, constraining long-term task performance in open-world Minecraft settings.

3. Method

In this section, we present Memo, a novel embodied agent designed to overcome the challenges of semantic cognition and inefficient open-world exploration. As illustrated in Fig. 1, Memo is built upon two core modules: 1) Multi-Scale Progressive Exploration, which plans efficient exploration trajectories through a coordinated global–regional–local structure, enabling both rapid coverage and fine-grained target search. 2) Dynamic Planning with Hierarchical Memory, which integrates fast, data-level short-term memory and slow, knowledge-level long-term memory. This allows the agent to rapidly retrieve recent experiences, associate high-level semantic knowledge, and dynamically reorder task priorities based on real-time observations. During operation, the exploration module proposes cross-scale exploration paths, while the hierarchical memory module continuously stores and retrieves environmental information. The dynamic planning module then fuses exploration objectives with memory feedback to adapt the agent's behavior, ensuring robust and efficient task execution in complex open-world environments.

CVPR #25998

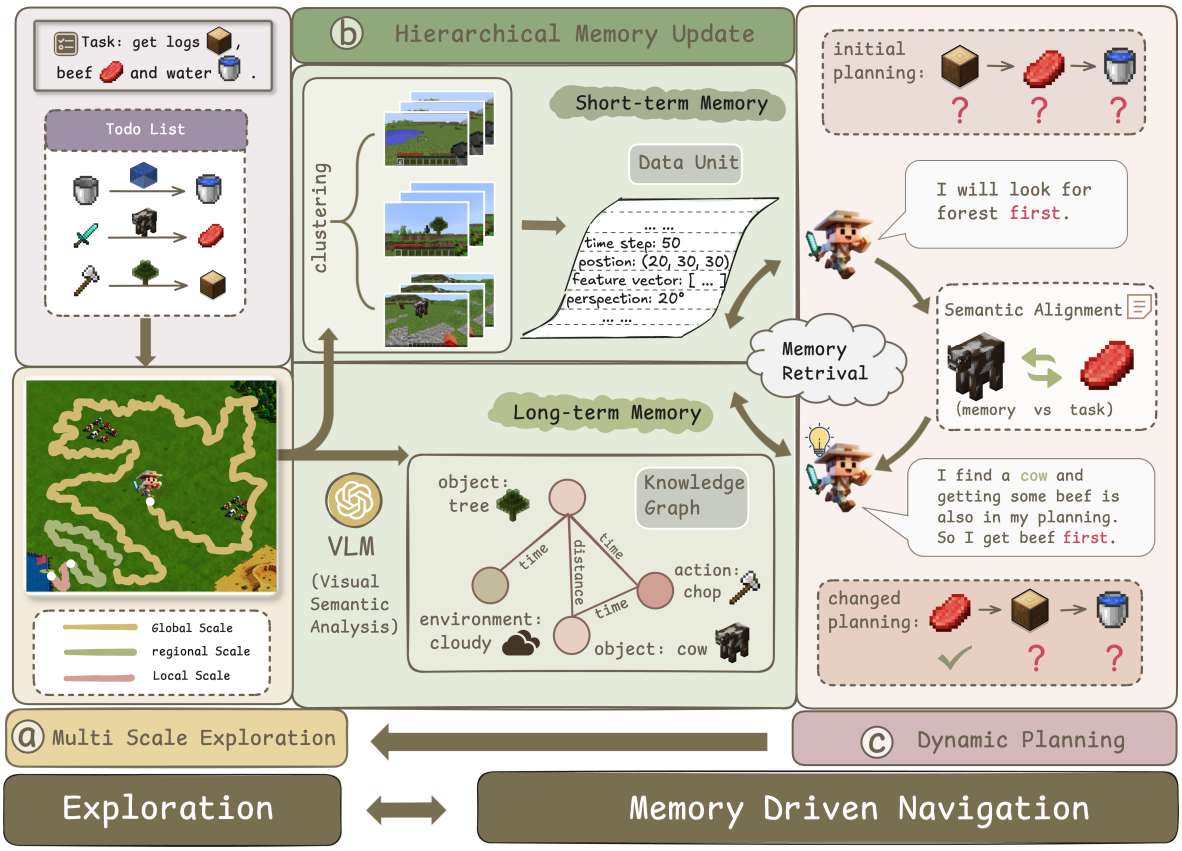


Figure 1. The overall architecture of Memo, which integrates exploration, memory, and planning for unified exploration–navigation. Tasks are decomposed into sub-tasks and executed through multi-scale progressive exploration: global rapid search, regional frontier refinement, and local execution. Newly observed information triggers hierarchical memory updates, storing short-term geometric cues for fast retrieval and long-term semantic knowledge in a graph. When the environment changes (e.g., encountering a cow while searching for a forest), Memo performs dynamic planning to reorder tasks and take advantage of new opportunities.

3.1. Multi-Scale Progressive Exploration

To overcome the inefficiency and locality issues of existing exploration strategies in large-scale environments, we propose a progressive exploration framework operating at the global, regional, and local scales. This framework balances exploration value and navigation cost, ensuring both wide-area coverage and fine-grained refinement.

3.1.1. Global-Scale Rapid Search

The global scale performs macro-level planning to identify promising exploration regions. Given Minecraft's effectively infinite world, we adopt an adaptive quadtree spatial index, where the root node covers the full map and child nodes represent candidate exploration regions. The quadtree depth and subdivision adapt dynamically based on visitation frequency. To select the next region, we compute a priority score:

$$Score(N_i) = \alpha \cdot Depth(N_i) + \beta \cdot Dist(P, N_i) \quad (1)$$

Where $Depth(N_i)$ is the depth of the current node, quantifying the retrieval efficiency of the node; $Dist(P, N_i)$ is the Euclidean distance of the path length from the current position P to N_i , quantifying the navigation cost. This method reduces the time complexity of spatial query from $O(n)$ to $O(logn)$, ensuring the efficiency of global planning.

3.1.2. Regional-Scale Frontier Refinement

At the regional scale, the agent focuses on frontier exploration, i.e., areas at the boundary between known and unknown space. Using a morphological-dilation-based frontier detector, we extract frontier points that maximize information gain. Each frontier point (f_i) is scored as:

$$Score(f_i) = \alpha \cdot Cover(f_i) + \beta \cdot Nove(f_i) + \gamma \cdot Entro(f_i) \quad (2)$$

Here, $Cover(f_i)$ is the area of the uncovered area within the visible range of the current viewpoint. $Nove(f_i)$ is the information entropy gain of the visible range of the current viewpoint. $Entro(f_i)$ is the distance between the current viewpoint and historical viewpoints.

CVPR #25998

3.1.3. Local-Scale Area Complement

To achieve a precise coverage of unexplored areas at a local level, we utilize a Voronoi region decomposition algorithm[11]. This algorithm divides the map into n Voronoi cells $\{\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_n}\}$, based on the visited set $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$ (where $v_i = (x_i, y_i, t_i)$, x_i and y_i are 2D coordinates, t_i is the timestamp).Each cell $\mathbf{C_i}$ is defined as:

$$\mathbf{C_i} = \{p \in \mathbf{\Omega} \mid \|p - v_i\|_2 < \|p - v_j\|_2,\ \forall j \neq i\} \quad (3)$$

Here, $\mathbf{\Omega}$ denotes the environmental space, and $\|\cdot\|_2$ represents the Euclidean distance. This definition ensures that the distance from any point p in cell $\mathbf{C_i}$ to the visited point v_i is smaller than its distance to other visited points, thus satisfying the local optimality of spatial division. Uncovered regions $\mathbf{U}$ can be automatically identified, where $\mathbf{U} = \mathbf{\Omega} \setminus \bigcup_{i=1}^{n} \mathbf{C_i}$ denotes the set difference operation, i.e., regions in $\mathbf{\Omega}$ that do not belong to any Voronoi cell $\mathbf{C_i}$. The scoring formula for the i-th unexplored region U_i is given by:

$$Score(U_i) = \alpha \cdot Area(U_i) + \beta \cdot Dist(P, U_i) \quad (4)$$

Where $Area(U_i)$ is the uncovered area, and $Dist(P, U_i)$ is the navigation cost. This drives the agent to cover large, nearby gaps.

We also implement a local escape mechanism to prevent stagnation. If the agent moves less than 5 blocks in 30 seconds, it replans locally. Repeated failure to reach a target triggers a target reset from memory, and repeated revisiting of the same location initiates a rollback to a key node. These mechanisms prevent local traps and improve exploration resilience.

3.2. Dynamic Planning with Hierarchical Memory Update

To overcome the limitations of fixed planning strategies, we introduce a hierarchical memory–driven dynamic planning framework. By integrating multi-level memory with adaptive decision-making, the agent can continuously refine its strategy based on both recent interactions and long-term semantic understanding, enabling flexible and efficient execution of sparse, long-horizon tasks.

3.2.1. Hierarchical Memory Update

Traditional memory systems rely heavily on raw data storage without semantic structuring, limiting their ability to support long-sequence reasoning. To address this, we design a hierarchical memory system: 1) Short-term memory provides rapid retrieval of recent experiences in terms of scene geometry. 2) Long-term memory constructs a structured semantic knowledge graph. This combination offers both real-time responsiveness and long-range associative reasoning.

Short-Term Geometric Memory for Rapid Experience Retrieval: The short-term memory focuses on processing and organizing recent interaction data to provide quick access to comparable past scenes. Built upon the Place Event Memory mechanism, it first extracts visual features from gameplay frames using MineClip [7]. These features are then clustered via the DP-Means algorithm [6], with positional information incorporated to distinguish event types more effectively. To handle the redundancy and real-time nature of Minecraft interactions, a sliding-window strategy is applied to automatically discard outdated experiences, ensuring that the memory pool preserves only high-value recent information.

Long-Term Semantic Memory for Knowledge Graph Construction: Given the computational cost of visual–language models (VLMs), we first perform keyframe selection to reduce redundant inputs while preserving semantically meaningful information. To this end, we employ a hybrid keyframe extraction strategy that combines global detection based on image entropy with local detection based on feature clustering. The global detector identifies major environmental changes by computing the entropy of each t frame I_t:

$$H(I_t) = -\sum_{i=0}^{255} p_i log_2 p_i \quad (5)$$

Here, p_i is the pixel intensity probability. If $|H(I_t) - H(I_{t-1}) > \theta_H|$ (where θ_H is the threshold), the frame is considered a global keyframe. Local detection local , based on feature clustering, captures new events in more detail. we use MineClip to extract current frame features and compare them with the historical feature library using cosine similarity. If the minimum cosine similarity is less than θ_c (where θ_c is the threshold), it is determined as a local keyframe. This dual detection can avoid missing important information and effectively filter duplicate frames.

Using the selected keyframes, we construct a long-term semantic memory graph with GPT-4o [16], converting discrete observations into structured and interpretable knowledge. The resulting graph consists of three fundamental node types: entity nodes (e.g., cows, stones, zombies), action nodes (e.g., mining, attacking, crafting), and environment nodes (e.g., darkness, rain, hunger states). Edges encode temporal relations (e.g., event sequences or traversal time between locations), spatial relationships (e.g., relative distances or co-occurrence), and causal dependencies. For instance, the entity nodes cow and beef are linked by a causal edge denoting "drops after killing." To construct the graph, we first extract structured "environment–object–action" triples from keyframes, and subsequently apply a pruning stage to remove redundant or low-relevance nodes. This process yields a lightweight yet se-

CVPR #25998

mantically expressive knowledge graph that supports long-horizon reasoning and interpretable memory retrieval.

3.2.2. Scene Semantics Alignment for Memory Retrieval

Memory retrieval in our framework proceeds through three sequential stages: query understanding, memory matching, and result ranking. To interpret user or task queries at a semantic level, we employ a pre-trained Word2Vec model to convert natural-language inputs into continuous semantic vectors. The model is trained on a Minecraft-specific corpus that comprehensively captures domain knowledge, including creature drops, item crafting dependencies, and container interactions. After tokenization and deduplication, the corpus is used to construct resource–relation–resource triples, resulting in a dense semantic space tailored to in-game concepts. This specialization enables accurate similarity estimation; for instance, the semantic closeness between "cow" and "beef" or between "water bucket" and "water" exceeds 97.9%.

Building on this semantic embedding space, memory retrieval integrates both short-term and long-term memory sources. Recent experiences are first queried from the short-term geometric memory to capture the most immediate contextual relevance. Subsequently, the query vector is compared against nodes in the long-term semantic memory graph, enabling retrieval of historical knowledge with deep semantic structure. Cross-referencing these two memory layers ensures that retrieval results balance immediate perceptual context and long-horizon semantic associations. Finally, all candidate results are jointly ranked according to semantic similarity and temporal relevance, ensuring that the retrieved memory is both contextually appropriate and temporally aligned with current task demands.

3.2.3. Dynamic Planning

The dynamic planning module serves as the integrative core that unifies perception, memory, and action, enabling the agent to adaptively regulate its behavior in complex environments. Inspired by human cognitive decision-making, this module operates through three tightly coordinated phases.

In the information fusion phase, the planner synthesizes heterogeneous inputs—including real-time environmental observations, exploration goals produced by the multi-scale exploration hierarchy, and semantic cues retrieved from memory—to form a holistic representation of the current decision context.

In the task priority sorting phase, pending subtasks are dynamically organized to avoid inefficiencies associated with static or predetermined execution orders. Each subtask is evaluated along three dimensions: urgency, semantic relevance (as informed by memory retrieval), and navigation cost. This multi-criteria prioritization ensures that the agent allocates resources effectively while maintaining responsiveness to evolving task requirements.

In the optimal action generation phase, the planner produces an action sequence that best satisfies the prioritized objectives under the current environmental conditions. By jointly considering task ordering, real-time constraints, and memory-derived semantic cues, the module closes the gap between strategic planning and low-level execution. This process enables coherent integration of exploration and task execution, effectively overcoming the disconnect that often arises in traditional controller-based systems.

4. Experiments

We evaluate the proposed Memo agent in sparse-object tasks within open-world environments along three dimensions: (i) large-scale exploration (Section 4.1), (ii) task-solving performance in sequential and non-sequential sparse tasks (Sections 4.2–4.3), and (iii) ablation studies to quantify the contributions of core modules (Section 4.4).

In open-world settings, critical resources are often sparsely distributed. For example, crafting a water bucket may require locating both water and iron across distant regions. Solving such tasks requires memory of resource locations and rational task sequencing. Accordingly, we design two primary task categories: sequential and non-sequential tasks.

4.1. Exploration Evaluation in Large-scale Environments

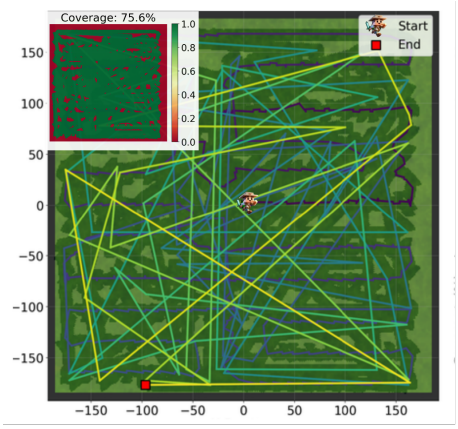
Experimental Setup Experiments are conducted in both simulated and real Minecraft environments. The simulated environment removes stochastic factors such as mobs and terrain variations to ensure reproducibility and isolate algorithmic performance. The real environment retains native interactions to validate practical applicability. Two map scales are used: small (128×128 blocks) for localized evaluation and large (384×384 blocks) for assessing open-world scalability.

We compare Memo against Steve-1 and MrSteve baselines, and a variant, Memo w/global, which only uses quadtree-based global planning. Steve-1 follows "Go Explore" instructions [2, 30], MrSteve combines count-based exploration with VPT-Nav [17], and Memo implements the proposed global–regional–local three-layer progressive exploration with VPT-Nav. Core metrics include map coverage rate and coverage efficiency.

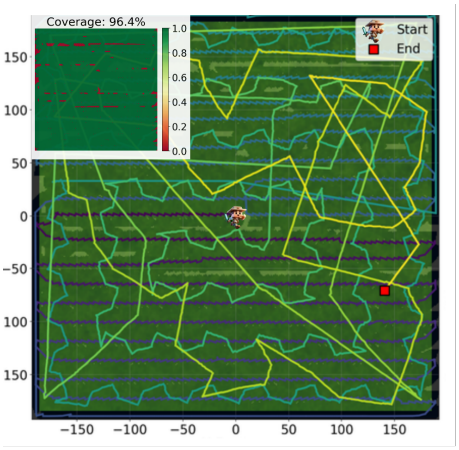
Results and Discussion Fig. 2 visually presents the exploration trajectories of `MrSteve` and `Memo` (ours) in a simulated environment. `MrSteve` frequently converges to local optima, leaving edges and key regions unexplored. In contrast, Memo's multi-layer exploration achieves more

complete coverage, reflecting effective coordination across global, regional, and local layers.

Tab. 1a and Tab. 1b reports coverage rates over time for MrSteve, Memo with global-only planning , and Memo across different map scales. Memo consistently achieves the highest coverage and efficiency. While global-only planning improves efficiency it can miss certain areas. Interestingly, as map size increases, MrSteve's coverage slightly improves because the proportion of missed edge regions decreases relative to the total map.



(a) Exploration trajectory of Mrsteve(baseline).



(b) Exploration trajectory of Memo(ours).

Figure 2. The trajectory of exploration over 10,000 timesteps in a simulated environment on a 384×384 map. Under the same conditions, Memo has more than 20% of map coverage than Mrsteve's and have smarter exploration strategy with more reasonable trajectory.

As shown in Tab. 1c , The results from real environments corroborate those from the simulated experiments. On the small real map , Memo's coverage rate reached 0.97 after 10,000 steps, significantly outperforming `Steve-1` (0.19) and `MrSteve` (0.67), with respective improvements of 78% and 30%. Similarly, on the large real map, Memo achieved a coverage rate of 0.83 after 40,000 steps, while

Method	128×128			
	500	1000	2000	6000
Mrsteve	0.46	0.57	0.60	0.59
Memo*(ours)	0.65	**0.87**	0.92	0.93
Memo(ours)	**0.66**	0.85	**0.93**	**0.99**

(a) Map coverage on 128×128 map in simulated environment.

Method	384×384			
	1000	4000	10000	20000
Mrsteve	0.17	0.53	0.75	0.83
Memo*(ours)	**0.19**	0.72	0.83	0.83
Memo(ours)	**0.19**	**0.73**	**0.96**	**0.98**

(b) Map coverage on 384×384 map in simulated environment.

Method	128×128		384×384	
	5,000	10,000	20,000	40,000
Steve-1	0.11	0.19	0.06	0.13
Mrsteve	0.64	0.67	0.39	0.69
Memo*(ours)	**0.72**	0.75	**0.44**	0.74
Memo(ours)	0.71	**0.97**	**0.44**	**0.83**

(c) Map coverage of different exploration policies in real environment.

Memo* means just use global level search when exploring. Map coverage $\in [0, 1]$. Best results in each column are highlighted in **bold**.

Table 1. Map coverage of different exploration policies. We use different steps on small (128×128) and large (384×384) maps to measure map coverage of different methods. From the result, we can see Memo outperform other methods typically when the scale of map and timestep is large. And when timestep is small, Memo without whole scale exploration, just using global level methods, is more effective.

baseline models struggled to effectively cover the extensive map due to their limited exploration efficiency. These experimental findings confirm that Memo's proposed exploration method is optimal, as `Steve-1` exhibits low map coverage and efficiency. Although `MrSteve`'s map coverage improved compared to `Steve-1`, it remains prone to missing locations and falling into local optima.

Furthermore, an independent evaluation of the global coverage layer revealed that, compared to `MrSteve`'s count-based target method, it yields a higher coverage rate and faster coverage efficiency. However, it still exhibits suboptimal exploration of local locations, underscoring the necessity of Memo's multi-scale approach. These results collectively demonstrate that Memo's progressive exploration framework effectively explores local positions and achieves more comprehensive coverage in complex, open-world environments.

CVPR #25998

4.2. Exploration and Navigation in Sequential Tasks

This experiment evaluates the agent's ability to solve sparse-resource, long-horizon tasks that require returning to previously visited locations. We design an ABA-Sparse task in which the agent must complete an A–B–A sequence—first locating resource A, then resource B, and finally returning to the initial A location. Successful execution therefore depends on efficient exploration to find scarce targets and reliable memory to revisit earlier sites.

Experimental Setup We compare three baselines—Steve-1, MrSteve, and a memory-only variant Memo*—against our full Memo agent. Memo* retains the hierarchical memory module but removes multi-scale exploration and dynamic planning. Performance is measured using task success rate and completion time. All experiments are conducted on a 128×128 sparse-resource map featuring ponds, forests, deserts, mountains, pits, and scattered ores that increase perception difficulty (Fig. 3).

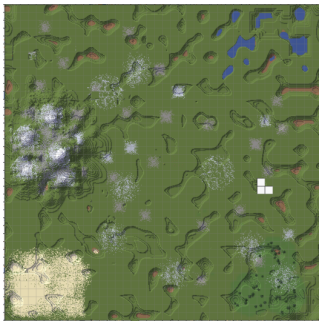


Figure 3. Real environment map in minecraft. Ponds are distributed in the upper right corner, forests in the lower right corner, high mountains on the left, and deserts in the lower left corner. Various minerals and deep pits are scattered across the map.

Results and Discussion As shown in Table 2, Steve-1 achieves a 0% success rate, largely due to its inability to locate sparse targets or resume tasks from past experiences. MrSteve performs better but still suffers from limited exploration and weak memory. Memo* further improves success rates, demonstrating the benefit of hierarchical memory alone.

The full Memo agent achieves the highest performance across all metrics. For example, in the scenario illustrated in Fig. 4, MrSteve fails to recognize the pond until repeated exploration increases its field-of-view coverage, only then enabling water retrieval. In contrast, Memo's VLM-enhanced semantic memory identifies and stores pond information immediately, enabling rapid return in the final "A" stage of the ABA sequence. While memory-only models help, their performance remains significantly below Memo's. This indicates that efficient completion of long-sequence tasks requires three components working in synergy: multi-scale exploration for rapid resource localization, hierarchical memory for accurate long-term recall, and dynamic planning for optimal execution order. Together, these modules address the core challenges of resource sparsity, unreliable recall, and weak task continuity, enabling robust performance in sequential and non-sequential sparse-resource tasks.

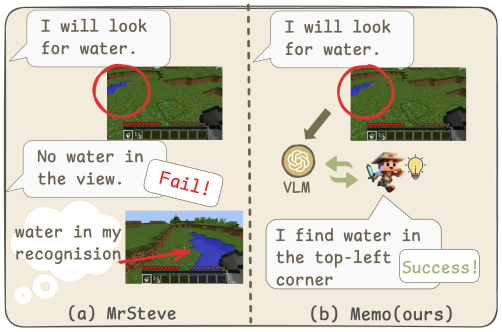


Figure 4. Exploration comparison in ABA sequence task. The left shows that MrSteve(baseline), based on PEM, fails to recognize the pond when view is limited. It only identifies the pond after its field of view is large enough. But Memo can recognize them both.

Method	Success Rate (%)	Average Timesteps
Steve-1	0	✗
Mrsteve	32	8123
Memo*	47	8093
Memo(ours)	**62**	**5981**

(a) ABA sequential task result

Method	Success Rate (%)	Average Timesteps
Steve-1	0	✗
Mrsteve	33	8040
Memo(ours)	**77**	**4620**

(b) ABC non-sequential task result

Memo* means without exploration.

Table 2. Result of success rates and average timesteps of different kinds of sparse task. Fig (a) represent the result of executing A task that has been performed historically. Memo is 1/4 faster than Mrsteve while achieving nearly twice success rate, indicating better utilization of historical information. Fig (b) also shows the efficiency of Memo compared to Mrsteve, because the sub-task sequency can be changed according to environment in our system.

4.3. Exploration and Navigation in Non-Sequential Tasks

To evaluate Memo's autonomous decision-making in more complex sparse-resource scenarios, we designed the ABC non-sequential task. Here, task element C may appear while

the agent is executing task A, requiring the agent to dynamically adjust task priorities based on environmental observations and historical memory rather than following a fixed sequence. This setup tests the agent's capacity for adaptive decision-making and resource coordination.

Experimental Setup We used the same baselines as in Section 4.2 and conducted experiments on a 128×128 sparse-resource map, consistent with the sequential task setup.

Results and Discussion As shown in Table 2b, Steve-1 fails completely (0% success), while MrSteve achieves a 33% success rate with an average completion time of 8040 steps, relying on limited memory and static exploration.

Memo substantially outperforms the baselines, achieving a 74% success rate and reducing the average completion time to 4620 steps. This improvement is enabled by the agent's dynamic planning module, which identifies that task C's resource lies along the current path during task A, dynamically reprioritizes tasks, and executes C first. By optimizing the sequence and leveraging multi-scale exploration for efficient resource localization, Memo minimizes redundant movements and task execution time. In contrast, MrSteve's fixed-sequence strategy leads to repeated searches and reduced efficiency, highlighting the critical role of adaptive planning in complex, open-world tasks.

4.4. Ablation Study

To assess the individual contributions of Memo's core modules, we conducted ablation experiments in the challenging non-sequential (ABC) task scenario on a 128×384 real map, using task success rate and average completion time as evaluation metrics (Tab. 3).

Method	Success Rates(%)	Timesteps
Memo w/o Memory	31	8034
Memo w/o Exploration	45	7942
Memo w/o Planner	62	6981
Memo(ours)	**77**	**4620**

Table 3. Results of ablation experiment.

The results clearly demonstrate the critical role of each module. Removing the hierarchical memory module (Memo w/o Memory) severely impairs the agent's ability to store and retrieve location information, reducing the success rate to 31% and nearly doubling task completion time. Excluding the multi-scale exploration module (Memo w/o Exploration) leads to inefficient local exploration and slower resource localization, yielding a 45% success rate and longer completion time. Without the dynamic decision-making module (Memo w/o Planner), the agent executes tasks in a fixed sequence, increasing path redundancy and resulting in a 62% success rate.

The full Memo model, integrating all modules, achieves the highest success rate (77%) and shortest completion time (4620 steps). As illustrated in Fig. 5, Memo dynamically reprioritizes tasks—for example, switching from "water-log-beef" to "beef-water-log" when encountering a cow—showcasing the synergistic operation of memory, exploration, and planning. These findings confirm that the hierarchical memory module provides accurate historical knowledge, multi-scale exploration ensures efficient resource localization, and dynamic planning enables optimal task scheduling, collectively forming Memo's core advantage in managing long-sequence tasks in open-world environments.

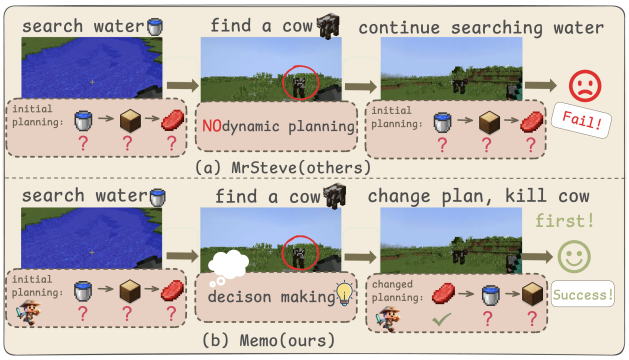


Figure 5. Plannig comparison in ABC's non-sequence. The initial task is 'water-log-beef'. When the Memo(ours) first find a cow, it switches its plan to 'beef-water-log' and prioritizes 'killing a cow' to obtain beef. However, due to the lack of dynamic planning ability, other agents can only continue searching for water.

5. Conclusion

In this paper, we presented Memo, a novel open-world agent featuring a hierarchical memory architecture for robust exploration, navigation, and task execution. Memo integrates three functional modules—explorer, memory, and planner—to dynamically coordinate perception, memory, and action, enabling effective completion of sparse sequence tasks in large-scale environments. Experimental results demonstrate that this integration significantly enhances task success and efficiency, highlighting Memo's capability to handle both environment- and memory-dependent challenges. Despite these advances, two limitations remain. First, reliance on GPT-4o incurs high computational costs, which could be mitigated with open-source alternatives. Second, our evaluation is currently limited to Minecraft, leaving cross-platform generalization as a promising direction for future work.

References

[1] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022. 2

[2] Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Groot: Learning to follow instructions by watching gameplay videos. *arXiv preprint arXiv:2310.08235*, 2023. 1, 5

[3] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, et al. Goat: Go to any thing. *arXiv preprint arXiv:2311.06430*, 2023. 2

[4] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020. 1, 2

[5] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12875–12884, 2020. 2

[6] Or Dinari and Oren Freifeld. Revisiting dp-means: fast scalable algorithms via parallelism and delayed cluster creation. In *Uncertainty in Artificial Intelligence*, pages 579–588. PMLR, 2022. 4

[7] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022. 4

[8] Jonathan Gray, Kavya Srinet, Yacine Jernite, Haonan Yu, Zhuoyuan Chen, Demi Guo, Siddharth Goyal, C Lawrence Zitnick, and Arthur Szlam. Craftassist: A framework for dialogue-enabled interactive agents. *arXiv e-prints*, pages arXiv–1907, 2019. 2

[9] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019. 2

[10] Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents. *arXiv preprint arXiv:2402.03610*, 2024. 2

[11] Jimmy Krozel and Dominick Andrisani. Navigation path planning for autonomous aircraft: Voronoi diagram approach. *Journal of Guidance, Control, and Dynamics*, 13 (6):1152–1154, 1990. 4

[12] Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *Advances in neural information processing systems*, 37:49881–49913, 2024. 2

[13] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36:69900–69929, 2023. 2

[14] Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, and Wei Yang. Juewu-mc: Playing minecraft with sample-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:2112.04907*, 2021. 1

[15] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR, 2017. 1

[16] R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5):1, 2023. 1, 4

[17] Junyeong Park, Junmo Cho, and Sungjin Ahn. Mrsteve: Instruction-following agents in minecraft with what-where-when memory. In *The Thirteenth International Conference on Learning Representations*, 2025. 1, 2, 5

[18] Yiran Qin, Enshen Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. Mp5: A multi-modal open-ended embodied system in minecraft via active perception. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16307–16316. IEEE, 2024. 2

[19] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. 1

[20] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023. 2

[21] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3009, 2023.

[22] Zhiyuan Sun, Haochen Shi, Marc-Alexandre Côté, Glen Berseth, Xingdi Yuan, and Bang Liu. Enhancing agent learning through world dynamics modeling. *arXiv preprint arXiv:2407.17695*, 2024. 2

[23] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017. 2

[24] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

[25] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 2

[26] Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong

Zheng, Yaodong Yang, et al. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2

[27] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53, 1998. 2

[28] Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. *arXiv preprint arXiv:2310.10021*, 2023. 2

[29] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34:25217–25230, 2021. 2

[30] Enshen Zhou, Yiran Qin, Zhenfei Yin, Yuzhou Huang, Ruimao Zhang, Lu Sheng, Yu Qiao, and Jing Shao. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*, 2024. 5

[31] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023. 2