MP5: A Multi-modal Open-ended Embodied System in Minecraft via Active Perception

Yiran Qin[1,2]*, Enshen Zhou[1,3]*, Qichang Liu[1,4]*, Zhenfei Yin[1,5],
Lu Sheng[3]†, Ruimao Zhang[2]†, Yu Qiao[1], Jing Shao[1]‡

[1]Shanghai Artificial Intelligence Laboratory [2]The Chinese University of Hong Kong, Shenzhen
[3]School of Software, Beihang University [4]Tsinghua University [5]The University of Sydney

yiranqin@link.cuhk.edu.cn zhouenshen@buaa.edu.cn
lsheng@buaa.edu.cn ruimao.zhang@ieee.org shaojing@pjlab.org.cn

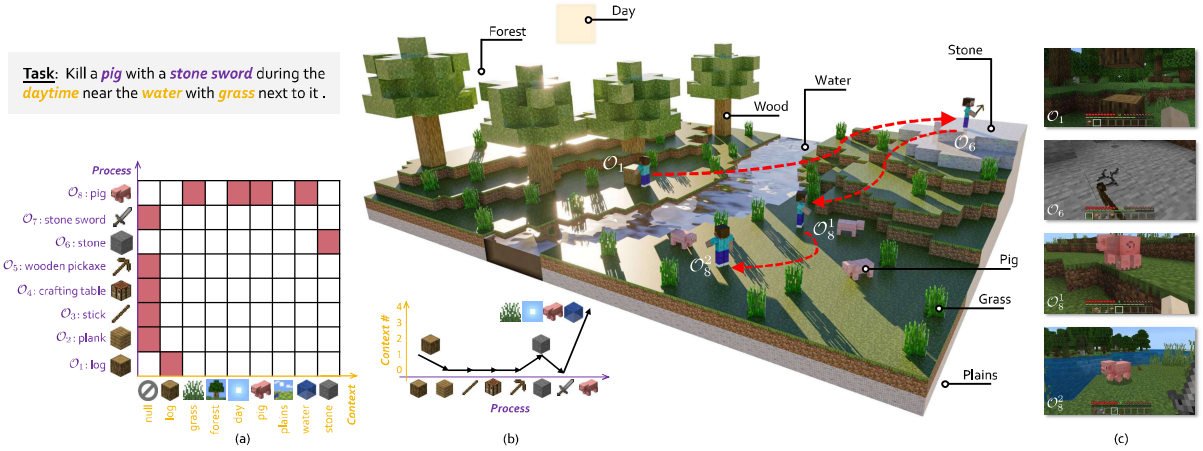

Figure 1. The process of finishing the task *"kill a pig with a stone sword during the daytime near the water with grass next to it."*. **(a)** To achieve the final goal (*i.e.*, $\mathcal{O}_8$: "kill a pig"), a player should accomplish a list of sub-objectives $\{\mathcal{O}_i\}_{i=1}^{7}$ sequentially. During this process, the player should also be aware of some items in the environment, *e.g.*, "grass", "day" and *etc.* **(b)** This diagram shows the number of these necessary items in the context that should be perceived for each sub-objective, during the task execution process. **(c)** Images marked by $\mathcal{O}_1$ and $\mathcal{O}_6$ show the observed ego-centric views in the process of achieving the corresponding sub-objectives. Images marked by $\mathcal{O}_8^1$ and $\mathcal{O}_8^2$ indicate how the player executes the action about the last sub-objective "kill a pig". This process tells that such long-horizon open-world embodied tasks in Minecraft should be solved both in the process-dependent and context-dependent way.

Abstract

It is a long-lasting goal to design an embodied system that can solve long-horizon open-world tasks in human-like ways. However, existing approaches usually struggle with compound difficulties caused by the logic-aware decomposition and context-aware execution of these tasks. To this end, we introduce MP5, an open-ended multimodal embodied system built upon the challenging Minecraft simulator, which can decompose feasible sub-objectives, design sophisticated situation-aware plans, and perform embodied action control, with frequent communication with a goal-conditioned active perception scheme. Specifically, MP5 is developed on top of recent advances in Multimodal Large Language Models (MLLMs), and the system is modulated into functional modules that can be scheduled and collaborated to ultimately solve pre-defined context- and process-dependent tasks. Extensive experiments prove that MP5 can achieve a 22% success rate on difficult process-dependent tasks and a 91% success rate on tasks that heavily depend on the context. Moreover, MP5 exhibits a remarkable ability to address many open-ended tasks that are entirely novel. Please see the project page at https://iranqin.github.io/MP5.github.io/.

* Equal contribution † Corresponding author ‡ Project leader

1. Introduction

One of the core objectives of current embodied intelligence is to construct generalist agents that can solve long-horizon open-world embodied tasks, approaching the behavior patterns of human beings [1, 16, 19, 27]. However, the *process dependency* and *context dependency* in these tasks, such as those in Minecraft depicted in Fig. 1, hinder recent agents from achieving the aforementioned goal. To be specific, the former emphasizes the inherent dependency among the sub-objectives of one task or an action sequence to fulfill one sub-objective (such as "craft a stone sword" should be solved before "kill a pig"). The latter highlights that the execution of each sub-objective or even each action depends on the contextual information of the environment (such as "kill a pig" requires to find the target "pig" and its surrounding items "grass" and "water" during the "daytime" in the observed images, as shown in Fig. 1).

The recent success of Large Language Models (LLMs) has attempted to solve the process-dependent challenge, by using LLMs to break down a long-horizon process-dependent task into a sequence of feasible sub-objectives [30, 31, 37]. These methods [30, 37] simplify the context-dependent challenge by assuming the agents are all-seeing, *i.e.*, knowing everything about their state and the environment it locates in. However, to solve the context-dependent challenge, an embodied agent should additionally have: (1) the perception capability is open-ended, selective and give results tailored to diverse purposes (*e.g.*, for task planning or action execution), (2) the perception module can be compatibly scheduled along with the other modules (*e.g.*, planning and execution modules) by a unified interface, as an integrated system.

To this end, we introduce *MP5*, a novel embodied system developed within Minecraft, to meet the above expectations. Specifically, *MP5* comprises five interacting modules, *i.e.*, **Parser** decomposes a long-horizon task into a sequence of sub-objectives that should be completed one by one; **Percipient** answers various questions about the observed images, as the reference for the other modules; **Planner** schedules the action sequences of a sub-objective, as well as refines the following sub-objectives, given the current situation; **Performer** executes the actions along with frequent interaction with the environment; and **Patroller** checks the responses from the Percipient, Planner, and Performer, for the purpose of verifying current plans/actions, or feedback on potential better strategies. In our work, Percipient is a LoRA-enabled Multimodal LLM (MLLM). Among the pre-trained LLMs, Parser and Planner are augmented with external Memory, while Patroller is not.

Notably, *MP5* includes an *active perception* scheme by means of multi-round interaction between Percipient and Patroller, which is to actively perceive the contextual information in the observed images, with respect to various queries raised by Planner and Performer. It is the key enabler to solve context-dependent tasks. Patroller in this scheme relays compatible feedback to Planner and Performer accordingly, while eventually strengthening the planning skill in awareness of the situations and improving the action execution correctness in an embodied manner.

Extensive experiments prove that *MP5* can robustly complete tasks needed for long-horizon reasoning and complex context understanding. It achieved a 22% success rate on diamond-level tasks (*i.e.*, one of the hardest long-horizon tasks) and a 91% success rate on tasks requiring complex scene understanding (*i.e.*, need to perceive around $4 \sim 6$ key items in the observed images). Moreover, in Sec. 4.2.3, *MP5* can surprisingly address more open-end tasks both with heavy process dependency and context dependency.

2. Related Work

2.1. Multi-modal Large Language Models

With the development of Large Language Models (LLMs) like the GPT series [2, 23, 25], as well as open-source LLMs such as the LLaMA series [28, 29] and Vicuna [5], Multi-modal Large Language Models (MLLMs) have emerged. Examples of such MLLMs include LLaVA [18], InstructBLIP [6], and LAMM [33], among others [4, 9, 14, 24, 32, 36]. In this work, we introduce MineLLM, which is specifically designed and trained for Minecraft, and leverage its perception, interaction, and analysis capabilities to build up Percipient for *MP5*, and further enable an objective-conditioned active perception scheme.

2.2. Agents in Minecraft

Previous works[3, 7, 8, 10, 16, 19, 34, 35] attempt to use approaches such as hierarchical RL, goal-based RL, and reward shaping to train an agent in Minecraft. MineCLIP [8] enables the resolution of various open-ended tasks specified in free language, even without any manually designed dense rewards. DreamerV3 [11] succeeds in training agents in Minecraft with a learned world model. VPT [1] builds a foundation model for Minecraft by learning from massive videos. Based on VPT, Steve-1 [15] also explores bringing in MineCLIP [8] to get an instruction following policy with high performance. The development of recent large language model-related work Voyager [30], DEPS [31], GITM [37] further promote the advancement of agents in long-horizon tasks. These works use pre-trained large language models as the zero-shot planners[13] for agents, leveraging the powerful reasoning capabilities of large language models to obtain continuous operation instructions or executable policy lists.

We take advantage of the reasoning capability of LLM to build up our own agent. Existing LLM agents [37, 37] in Minecraft feed scene data from simulation platforms [8, 10]

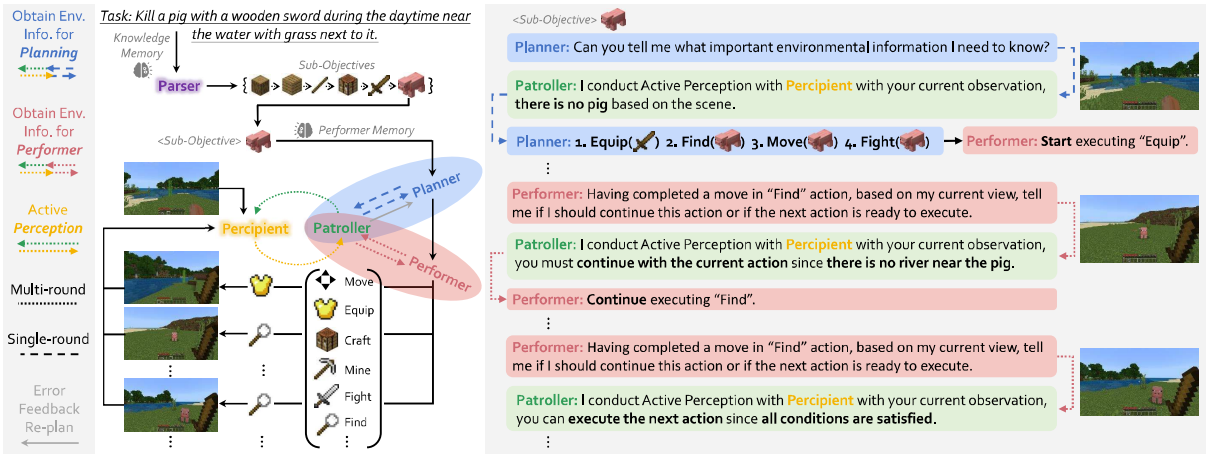

Figure 2. Overview of module interaction in *MP5*. After receiving the task instruction, *MP5* first utilizes Parser to generate a sub-objective list. Once a sub-objective is passed to the Planner, the Planner Obtaining Env. Info. for Perception-aware Planning. The performer takes frequently Perception-aware Execution to interact with the environment by interacting with the Patroller. Both Perception-aware Planning and Execution rely on the Active Perception between the Percipient and the Patroller. Once there are execution failures, the Planner will re-schedule the action sequence of the current sub-objective. Mechanisms for collaboration and inspection of multiple modules guarantee the correctness and robustness when *MP5* is solving an open-ended embodied task.

into large language models for task planning. However, for embodied agents in real scenes, it is clearly unrealistic to use accurate scene data directly. Therefore, agents need to be robust to make decision corrections despite inaccurate or erroneous perception information. Moreover, open-ended tasks need hierarchical reasoning [19] and complex open-ended context understanding [1, 8], classical perception networks can only output fixed perception results and cannot provide corresponding perception information according to the task, making it impossible to understand open-ended scenarios. Therefore, we design *MP5*, an embodied agent with open-ended capabilities that can solve the problem of open-ended tasks.

3. Method

In this section, we first give an overview of our proposed *MP5*, for solving context-dependent and process-dependent tasks in an open-world and embodied environment, such as Minecraft (Sec. 3.1). Next, we elaborate on how to implement an active perception scheme (Sec. 3.2). This scheme plays a vital role in *MP5* to solve context-dependent tasks, since it reliably grounds the visual content according to different kinds of objectives, and thus strengthens the planning skill and execution correctness with respect to context-dependent tasks. Then, we show how to plan and update action sequences in awareness of the situations, and how to reliably execute these actions in an embodied environment (Sec. 3.3). Finally, we give necessary implementation details about *MP5* in Sec. 3.4.

3.1. Overview

As demonstrated in Fig. 2, our *MP5* includes five major modules, *i.e.*, Parser, Percipient, Planner, Performer, and Patroller. Specifically, Percipient is a parameter-efficiently fine-tuned Multimodal Large Language Model (MLLM) that is specified to the Minecraft environment. The Parser, Planner, and Patroller are pre-trained Large-language Models (LLMs). We also include retrieval-augmented generation (RAG) to enhance the quality of responses generated by Parser and Planner. Performer is an interface that explains each action from the action sequence into executable commands that directly control the game character.

Why can *MP5* solve context-dependent and process-dependent tasks? *MP5* includes an *active perception* scheme by means of multi-round interactions between Percipient and Patroller, which is to actively perceive the environmental information in the observed images, with respect to various objectives raised by Planner or Performer. With the help of this scheme, Planner can schedule or update action sequences in awareness of the observed images, inventory status and *etc*., resulting in a *situation-aware planning*; Performer can execute actions that are adapted to the embodied environment, resulting in a *embodied action execution*. Patroller in this scheme can also feedback on better choices of plans/actions based on the visual evidence so that the process-dependent tasks are solved with fewer chances of context-dependent execution failures. Moreover, Percipient can understand open-ended visual concepts, therefore it allows *MP5* to solve tasks that are never seen before.

How does *MP5* function? In Fig. 2, upon receiving a

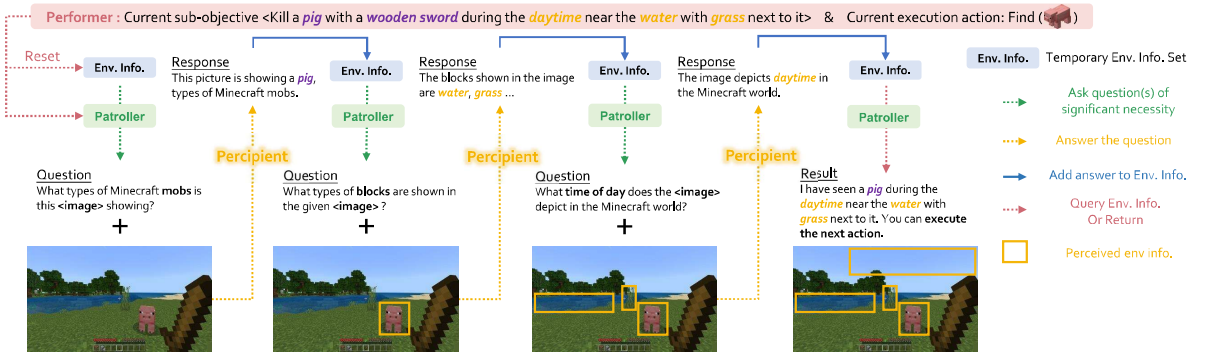

Figure 3. A demonstration of the process of Active Perception scheme. Temporary Env. Info. Set saves information collected in the current scenario, so it should be reset at the beginning of Active Perception scheme. Performer then invokes Patroller to start asking Percipient questions with respect to the description of the sub-objective and the current execution action round by round. The responses of Percipient are saved in Temporary Env. Info. Set and are also gathered as the context for the next question-answering round. After finishing asking all significant necessary questions, Patroller will check whether the current execution action is complete by analyzing the current sub-objective with Perceived env info. saved in Temporary Env. Info. Set, therefore complex Context-Dependent Tasks could be solved smoothly.

high-level task, *MP5* first utilizes the Parser to generate a sequence of short-horizon sub-objectives, as a list of rich instructions in natural languages. The feasibility of the generated sub-objectives is augmented by retrieving an external Knowledge Memory. This knowledge mainly comes from three sources: part of it is from the online wiki, another part is from the crafting recipes of items in MineDojo [8], and some are user tips from Reddit. To one sub-objective, Planner schedules the action sequence that is grounded by the environmental information gathered by the active perception scheme. In this case, Performer will execute the actual actions by explaining the action sequence that is adapted to the embodied environment, via frequent interaction with the active perception scheme. Once there are execution failures (determined by Patroller), Planner will re-schedule the action sequence of the current sub-objective, or even update the following sub-objectives if some necessary sub-objectives are missing. Otherwise, the agent will go to the next sub-objective and schedule new action sequences, whilst the successful action sequence of the current sub-objective will be stored in the external memory of Planner (called Performer Memory), along with the agent situation when it was planned. In the end, the agent will stop when the final sub-objective of the task has been reached.

3.2. Active Perception

Let's take the example shown in Fig. 3 to demonstrate how the active perception scheme works. In this example, the active perception scheme is communicated with Performer to enable an embodied action execution.

At first, Performer invokes Patroller to start asking Percipient questions with respect to the description of the sub-objective and the current execution action, while simultaneously resetting the set of environmental information to be gathered. Then Patroller progressively asks Percipient whether the observed image contains necessary items/factors (*e.g.*, mobs  , blocks  , time) related to recent sub-objective (*e.g.*, pig ) and the executing action (*e.g.*, "find pig "). The responses of Percipient are also progressively gathered and act as the context for the next question-answering round. Note that in each round, Patroller also checks whether all the necessary items/factors have been collected - If yes, Patroller stops the interaction and returns all the environmental information as natural language, and invokes Performer to execute the next action. If Patroller eventually fails to gather enough items/factors, it will tell Performer what items/factors are missing in the observed images, which suggests Performer keeps executing the current action. Please also check the example shown in Fig. 2.

Similarly, active perception used in situation-aware planning is similar to what is explained here, except that the applied instructions do not contain the executable action. For more details please check the Sup. E.

3.3. Perception-aware Planning and Execution

Situation-aware Planning. Given one sub-objective, Planner will generate the action sequence based on the description of the situation, such as the objective-conditioned environmental information from the active perception scheme, the inventory status and localization, and *etc*. Moreover, Planner will retrieve previous successful action sequences as the demonstration prompt to augment the aforementioned planning results. If the active perception scheme fails to find the key items/factors about the current sub-objective in the observed image, the generated action sequences will include more actions to reach them. Moreover, if Performer encounters execution failures determined by Patroller (such

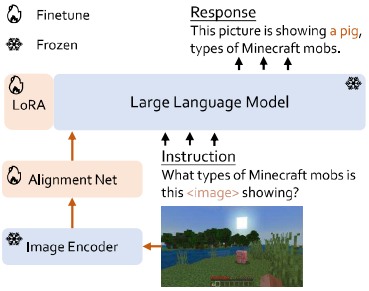

Figure 4. The model architecture of MineLLM. Image is encoded by a pre-trained vision encoder and decoded by LLM. Only the parameters of Alignment Net and LoRA are trainable.

as failure of "equip wooden sword"), Planner will reschedule the action sequence or even update the following sub-objectives, with the help of external memories.

Embodied Action Perception. As indicated in Sec. 3.2, Performer would like to communicate with the active perception scheme in every round of action execution, so as to enhance the ego-centric awareness of the agent. The new action will be executed if Patroller identifies necessary environmental information in the observed images that matches both the sub-objective and the goal of the current action. Otherwise, the current action is kept executing until encountering execution failures or the end of the episode. The successful action sequence about one sub-objective will be stored in the Performer Memory, together with necessary situational information of the agent when it was planned. For more details about the planning and execution process, please check Sup. G.2 and Sup. B.2.

3.4. Implementation Details

Percipient. The network of Percipient is depicted in Fig. 4. Images are processed by a frozen vision encoder MineCLIP [8], whose features are projected by an Alignment Net(we use two-layer MLP like LLaVA-1.5 [17]) to the same feature space as the text embeddings of the applied LLM (we use Vicuna-13B-v1.5 [5]). Then the vision and text tokens are concatenated to feed into a LoRA-based fine-tuned LLM [12]. We add LoRA [12] parameters to all projection layers of the self-attention layers in the LLM. Only the parameters of the Alignment Net and the LoRA module are optimized during training. The construction of the training data with respect to Percipient is in the Sup. B.1.

Parser, Planner, and Patroller. We utilize OpenAI's GPT-4 [22] as LLMs in Parser, Patroller, and Planner. We also evaluate other alternatives of GPT-4 [22], such as open-source models like Vicuna-13B-v1.5 [5] and LLaMA2-70B-Chat [29] in Sup.D.3.

Performer. It is important to clarify that the actions generated by Planner are not low-level commands such as keyboard and mouse operations [1], but a set of simple actions (such as equip, move, craft). Inspired by GITM [37], we implement these actions appropriately through basic operations provided by the MineDojo [8] simulator. For more details, please check the Sup. B.2.

4. Experiments

At first, we depict the setup of the Minecraft simulation environment that we build and validate *MP5*, and give the definition of the evaluated tasks and how to set them in Sec. 4.1. In Sec. 4.2, we present the quantitative and qualitative performance of *MP5*, as well as in-depth discussions on these tasks, and demonstrate that *MP5* can even successfully accomplish tasks that are more open-ended and never seen before. At last, we investigate how different modules affect the performance of *MP5* and analyze the impact of various module choices within our system in Sec. 4.3.

4.1. Experimental Setup

Environment Setting. We employ MineDojo [8] as our simulation environment to build and validate *MP5*. We capture player ego-view images provided by MineDojo [8] as input of *MP5*, and further construct a dataset for training MineLLM. As for the output of *MP5*, we encapsulate MineDojo's [8] actions to create our own action space.

Task Setting. To evaluate how our *MP5* can integrate perception information with planning and execution, we define two types of tasks: *Context-Dependent Tasks* and *Process-Dependent Tasks* as illustrated in Tab. 1 and Tab. 2.

1) Context-Dependent Tasks primarily study how Active Perception enables the agent to better perceive low-level context information in the environment. We first establish 6 aspects of environmental information derived from the Minecraft game environment: *[Object, Mob, Ecology, Time, Weather, Brightness]*. Each aspect has multiple options. For example, pigs, cows, and sheep are all elements belonging to Mob. Based on this, we define 16 tasks and organize their difficulty into four levels by taking into account the number of information elements that require perception, as is shown in Tab. 1. For example, Easy tasks necessitate the perception of only one element, whereas Complex tasks involve the perception of 4 to 6 elements. We rigorously assess *MP5*'s proficiency in environmental context perception across these 16 tasks. In *Context-Dependent Tasks*, our environment details are predetermined (*e.g.*, biomes, weather, and *etc.*), as certain targets are exclusive to specific environments. Without this environmental specificity, the agent might never encounter the intended target. We retain each observation of active perception throughout the task, using them as references to ascertain the agent's successful completion of the task.

2) Process-Dependent Tasks focus on exploring the con-

Table 1. *Context-Dependent Tasks*. 16 tasks are defined and divided into 4 difficulty levels based on the minimum number of information types needed. Underlines label the environmental information, reflecting the complexity varies at each level.

Task Level	Example Task
Easy	Find a tree
Mid	Find a tree in the forest
Hard	Find a tree in the forest during the nighttime
Complex	Find a pig near a grass in the forest during the daytime

Table 2. *Process-Dependent Tasks*. 25 tasks are defined and divided into 5 difficulty levels based on increasingly increasing reasoning steps. A higher difficulty level implies that the agent needs to engage in longer reasoning and planning.

Task Level	Reasoning Step	Example Task
Basic	1-3	craft crafting table
Wooden	4-5	craft wooden sword
Stone	6-9	mine stone
Iron	10-11	smelt iron ingot
Diamond	>11	obtain diamond

tributions of situation-aware planning, embodied action execution, and the integration with Active Perception in accomplishing long-term tasks while constantly perceiving the environment and dynamically adjusting actions. We select 25 tasks from the technology tree and define their difficulty levels as Basic level to Diamond level based on the number of reasoning steps required to complete the tasks. All environmental factors (*e.g.*, biomes, weather, and *etc.*) are randomized in *Process-Dependent Tasks*. More details can be found in Sup.D.1.

Evaluation Metrics. For different tasks, the agent's initial position and environment seed are randomized. The agent begins in survival mode, commencing with an empty inventory, and faces the challenge of hostile mob generation. It starts from scratch, with a game time limit of 10 minutes, a time period equivalent to 12,000 steps at a control frequency of 20Hz. More details can be found in Sup. C.

For the *Context-Dependency Tasks*, each assignment is open-ended. Therefore, we conduct manual evaluations when the agent determines it has completed the task or exceeds the time limit. Two cases are ruled as failures: 1)There is an observation that meets all the conditions, but the agent does not end the task; 2) The last observation does not meet all the conditions, yet the agent ends the task. Otherwise, we believe that the agent correctly perceives all the context according to the task and determines that the task is successfully completed. For the *Process-Dependent Tasks*, any accidental deaths of the agent during the game are counted as failures, as are instances where the agent does not accomplish the task within the time limit.

In practice, we conduct 50 games on *Context-Dependent Tasks* and 30 games on *Process-Dependent Tasks*, averaging the success rates for both. The results are grouped according to the previously defined difficulty levels, and report the group means. For detailed definitions of the evaluation, please refer to Sup. D.

4.2. Main Results

4.2.1 Results of Context-Dependent Tasks

In *Context-Dependent Tasks*, we primarily investigate how to enhance an agent's perception of context information within the environment. We demonstrate the performance difference between Active Perception and other perception methods. We compare them with pre-trained multimodal large language models LLaVA-1.5 [17] and GPT-4V [21], and analyze the performance of both active and fine-grained global perception on the tasks in Tab. 3. Although fine-grained global perception can obtain comprehensive perceptual information, due to the lack of objective-conditioned attention, the objective-related information obtained may be lacking or incorrect. Active perception only focuses on objective-related information and ignores other useless information, so that more accurate objective-related information can be obtained and better performance in *Context-Dependent Tasks* can be achieved. For the comparison, we use MineLLM, which is fine-tuned on the Minecraft instruction dataset we collect, slightly better than GPT-4V [21], which is trained on massive data, and substantially better than LLaVA-1.5 [17], which is not fine-tuned on instruction data. The complete results of *Context-Dependent Tasks* can be found in Sup.D.2.

4.2.2 Results of Process-Dependent Tasks

In *Process-Dependent Tasks*, we report the performance of the agent in completing long-horizon tasks by continuously perceiving the environment context and dynamically adjusting its actions. We also investigate the agent's behavior in scenarios of non-situation-aware planning and non-embodied action execution. The complete results of *Process-Dependent Tasks* can be found in Sup.D.2.

Considered the landscape of related works [1, 11, 30, 31, 37], we refrain from making direct comparisons due to the substantial variations in the **observation space**, **action space**, **environmental setup**, and **game termination conditions**. Notably, VPT [1] emulates human players' keyboard and mouse controls, DreamerV3 [11] is trained from scratch for diamond collection in a modified Minecraft environment with altered block-breaking mechanics using world models, DEPS [31] integrates LLM planning and a learning-based control policy based on MineDojo [8] actions, GITM [37] employs privileged information such as lidar perception, and Voyager [30] utilizes purely text-based

Table 3. Performance on *Context-Dependent Tasks*. We compare the success rate of different Methods and different Perception strategies. We set up special prompt to make the output of the caption as comprehensive as possible, this perception method is called Fine-Grained Global Perception. We use A to denote Active Perception, and G to denote Fine-Grained Global Perception.

Method	Strategy	Average Success Rate(%)			
		Easy	Mid	Hard	Complex
LLaVA-1.5 [17]	G	47.5	22.5	5.0	0.0
	A	72.5	50.0	11.0	0.0
GPT-4V [21]	G	97.5	85.0	75.0	60.0
	A	**100.0**	94.5	92.5	87.5
***MP5*(Ours)**	G	90.0	82.5	77.5	67.5
	A	98.5	**94.5**	**93.0**	**91.0**

information perception in collaboration with the Mineflayer API for action. It is crucial to note that in Voyager [30], items are not dropped upon the agent's death. Given that our experiments aim to showcase the system's capability to adapt both process-dependent reasoning and complex context-understanding tasks, our focus turns to presenting the following two key insights:

Embodied action execution is critical for open-ended tasks. Comparing *MP5* w/o E. and *MP5* in Tab. 4, we can observe that when an agent is unable to interact with the environment and access low-level environment contextual information during action execution, it essentially becomes "blind", unable to determine the termination of its actions based on environment. Therefore, the success rate in *Process-Dependent Tasks* is 0.00%.

Situation-aware planning leads to more scenario-appropriate strategies. Comparing *MP5* w/o P. and *MP5* in Tab. 4, we observe that the lack of environment contextual information during the agent's planning process can lead to erroneous or redundant actions, thereby reducing the success rate (for example, the success rate in diamond-level tasks decrease from 22.00% to 14.00%). Consider a scenario where the current sub-objective is "kill a pig". If a pig is already present, the agent should directly execute "move" to approach without the need to first "find" then "move". However, the relatively small decrease in the success rate can be attributed to the dynamic adjustment of perception and action execution offered by embodied action execution. Simultaneously, when errors are detected, the perceived environmental information and the erroneous actions can be fed back to the planner for re-planning.

4.2.3 Open-Ended Tasks

Processing long-horizon reasoning and understanding complex contexts are interconnected in the real world. For simplicity and comparability of the experimental setup, the first two task settings do not consider the intersection of process and context, as we cannot exhaust all combinations

Table 4. Performance on *Process-Dependent Tasks*. We compare the success rate when interacting or not with the environment during the planning or execution. w/o P. and w/o E. indicates non-situation-aware planning and non-embodied action execution.

Method	Average Success Rate(%)				
	Basic	Wooden	Stone	Iron	Diamond
MP5 w/o P.	0.00	0.00	0.00	0.00	0.00
MP5 w/o E.	92.00	86.00	68.67	45.33	14.00
MP5	96.00	88.67	76.00	52.00	22.00

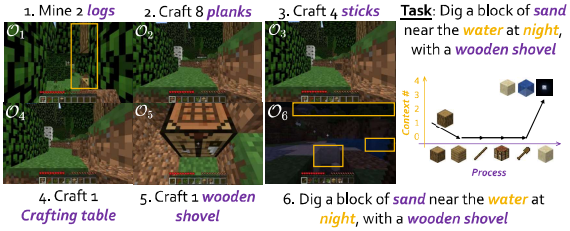

Figure 5. Screenshots of *"Dig a block of sand near the water at night with a wooden shovel"*. In *Open-Ended Tasks*, the agent needs to better integrate low-level context information and high-level decision-making, making it extremely challenging.

that these two task dimensions can form. Therefore, we refer to tasks that incorporate both Process-Dependent and Context-Dependent elements as *Open-Ended Tasks*. Specifically, these tasks require the agent to perceive different information of the environment at multiple stages of completing sub-objectives. As shown in Fig. 5, we present an example of an *Open-Ended Task*, named *"Dig a block of sand near the water at night with a wooden shovel"*. We conduct extensive validations on this type of task, proving that *MP5* can complete long-sequential tasks in challenging environments. More demonstrations and experimental results of *Open-Ended Tasks* can be found in Sup.F.3.

4.3. Ablation Study

We conduct ablation studies to evaluate the effectiveness of various modules. The experimental setup and the associated success rates are in Sec. 4.1. More detailed ablation studies are listed in Sup.D.3. The following paragraphs present the analyses derived from our ablation studies.

Model pre-trained on massive data of Minecraft can better comprehend the Minecraft appearance styles. We conduct ablation studies on the multi-modal large language model (MLLM) part within *Context-Dependent Tasks* in Tab. 5, comparing the performance outcomes of different MLLMs and different pre-trained visual encoders in the percipient. We find the performance of the open-source model LLaVA-1.5 [17] to be relatively weak, with a success rate of merely 50.00% at the Mid level and 11.00% on the Hard level. This is primarily due to the model's train-

Table 5. Success rates for different MLLMs and pre-trained visual encoders in the percipient on *Context-Dependent Tasks*

Method	Visual Encoder	Average Success Rate(%)			
		Easy	Mid	Hard	Complex
LLaVA-1.5 [17]	CLIP [26]	72.50	50.00	11.00	0.00
MineLLM	CLIP [26]	95.00	90.00	87.00	80.00
MineLLM	MineCLIP [8]	98.50	94.50	93.00	91.00

Table 6. Success rates for different LLMs as zero-shot Planner on *Process-Dependent Tasks*

Planner	Average Success Rate(%)				
	Basic	Wooden	Stone	Iron	Diamond
Vicuna-13B-v1.5 [5]	1.33	0.00	0.00	0.00	0.00
GPT-3.5-turbo [20]	95.33	86.67	42.00	2.67	0.00
GPT-4 [22]	96.00	88.67	76.00	52.00	22.00

ing predominantly on real-world data, causing it to struggle with the pixel-style image recognition characteristic of Minecraft. We also discover that, when the visual encoder is frozen, the MineLLM with CLIP [26] as its visual encoder consistently performs worse across all levels compared to MineLLM with MineCLIP's [8] pre-trained single image visual encoder. It may caused by, in the case of a frozen visual encoder, a visual encoder pretrained on massive data of Minecraft can align with pixel-style images more rapidly.

Enhanced reasoning ability results in improved planning. We compare the performance of open-source large language models, OpenAI's GPT-3.5-turbo [20] in Tab. 6, and GPT-4 [22] as zero-shot Planners on *Process-Dependent Tasks*. We find that as the models' inferential capabilities increase, the Planner produces better results by planning in a situation-aware method, yielding more concise and accurate execution actions. The Vicuna-13B-v1.5 [5] model, when used as a Planner, struggles to produce effective plans, achieving only a 1.33% accuracy rate at the Basic level. GPT-4 [22] exhibits the best performance, attaining a 22.00% success rate at the Diamond level, whereas both Vicuna-13B-v1.5 [5] and GPT-3.5-turbo [20] score 0.00%.

Leveraging memory leads to better planning. In our Performer Memory, we store previously successful sub-objectives and their corresponding execution actions. When planning in similar scenarios, Performer Memory can provide the Planner with similar execution action plans for completing the sub-objectives. While the plans may not be identical, they can effectively assist the Planner in performing situation-aware planning. Comparing the first and last rows of Tab. 7, we find that without the Performer Memory, the success rate of tasks at all levels decreases (Diamond level drops from 22.00% to 16.67%). However, the decrease is not significant as the Performer Memory primarily serves a reference function, with specific action planning still heavily reliant on the Planner's capabilities.

Robustness is essential in open-world settings. To enhance the robustness evaluation of our system, we introduce

Table 7. Success rates on different modules within *Process-Dependent Tasks*: We study the roles of the Performer Memory (PM) and the check part of Patroller (P), with 'RD' denoting *"Random Drop"* setting. ✓denotes the inclusion of the module or setting, and ✗ indicates its absence.

PM	P	RD	Average Success Rate(%)				
			Basic	Wooden	Stone	Iron	Diamond
✗	✓	✗	96.00	87.33	67.33	47.33	16.67
✓	✗	✓	70.00	7.33	0.67	0.00	0.00
✓	✓	✓	87.33	76.67	45.33	18.67	1.33
✓	✓	✗	96.00	88.67	76.00	52.00	22.00

a *"Random Drop"* setting. In this setting, we randomly discard one complete sub-objective from the inventory at the start of each new sub-objective, which deliberately induces execution errors for the agent. Comparing the second and third lines in Tab. 7, we observe the critical role of the Patroller in recognizing feedback errors. The Patroller's ability to integrate current environmental information with error information is essential for enabling the planner to re-plan. The significance of this robustness is evident when examining the success rates. Without the Patroller's robustness, the agent's success rate on the Wooden level plummets from 76.67% to 7.33%, while success rates on the Iron, and Diamond levels drop to 0.00%. Details regarding the *"Random Drop"* setting can be found in Sup.D.3.

5. Conclusion

In this paper, we propose a novel multi-modal embodied system termed *MP5* which is driven by frequently egocentric scene perception for task planning and execution. In practice, it is designed by integrating five functional modules to accomplish task planning and execution via actively acquiring essential visual information from the scene. The experimental results suggest that our system represents an effective integration of perception, planning, and execution, skillfully crafted to handle both context- and process-dependent tasks within an open-ended environment.

Limitation and Future Work. Two major limitations need to be clarified. Firstly, the reliance on GPT-3.5-turbo [20] or GPT-4 [22] limits the system's usability, as not everyone has access to these APIs. Secondly, the scope of the applied simulation platform is limited. Despite showing promising performance in Minecraft, we haven't extended our exploration to other simulation platforms, which is a potential area for further research.

Acknowledgement. This work was supported by the National Key R&D Program of China (2021YFB1714300), the National Natural Science Foundation of China (62106154, 62132001), the Natural Science Foundation of Guangdong Province, China (2022A1515011524).

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, 3, 5, 6

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

[3] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13734–13744, 2023. 2

[4] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023. 2

[5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023. 2, 5, 8

[6] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023. 2

[7] Ziluo Ding, Hao Luo, Ke Li, Junpeng Yue, Tiejun Huang, and Zongqing Lu. Clip4mc: An rl-friendly vision-language model for minecraft. *arXiv preprint arXiv:2303.10571*, 2023. 2

[8] 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. 2, 3, 4, 5, 6, 8

[9] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023. 2

[10] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: a large-scale dataset of minecraft demonstrations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2442–2448, 2019. 2

[11] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023. 2, 6

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 5

[13] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022. 2

[14] Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, et al. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. *arXiv preprint arXiv:2312.06739*, 2023. 2

[15] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *arXiv preprint arXiv:2306.00937*, 2023. 2

[16] 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. 2

[17] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023. 5, 6, 7, 8

[18] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 2

[19] 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. 2, 3

[20] OpenAI. Introducing chatgpt. 2022. 8

[21] OpenAI. Gpt-4v(ision) system card. 2023. 6, 7

[22] R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023. 5, 8

[23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022. 2

[24] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023. 2

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 8

[27] 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. 2

[28] 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. 2

[29] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2, 5

[30] 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, 6, 7

[31] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multitask agents. *arXiv preprint arXiv:2302.01560*, 2023. 2, 6

[32] Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*, 2023. 2

[33] Zhenfei Yin, Jiong Wang, Jianjian Cao, Zhelun Shi, Dingning Liu, Mukai Li, Lu Sheng, Lei Bai, Xiaoshui Huang, Zhiyong Wang, et al. Lamm: Language-assisted multimodal instruction-tuning dataset, framework, and benchmark. *arXiv preprint arXiv:2306.06687*, 2023. 2

[34] Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2023. 2

[35] 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. 2

[36] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 2

[37] 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 enviroments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023. 2, 5, 6