See and Think: Embodied Agent in Virtual Environment

Zhonghan Zhao[1,♠], Wenhao Chai[2,♠], Xuan Wang[1,♠], Boyi Li[1], Shengyu Hao[1], Shidong Cao[1], Tian Ye[3], and Gaoang Wang[1,†]

[1] Zhejiang University
[2] University of Washington
[3] Hong Kong University of Science and Technology (GZ)

Abstract. Large language models (LLMs) have achieved impressive progress on several open-world tasks. Recently, using LLMs to build embodied agents has been a hotspot. This paper proposes STEVE, a comprehensive and visionary embodied agent in the Minecraft virtual environment. STEVE comprises three key components: vision perception, language instruction, and code action. Vision perception involves interpreting visual information in the environment, which is then integrated into the LLMs component with agent state and task instruction. Language instruction is responsible for iterative reasoning and decomposing complex tasks into manageable guidelines. Code action generates executable skill actions based on retrieval in skill database, enabling the agent to interact effectively within the Minecraft environment. We also collect STEVE-21K dataset, which includes 600+ vision-environment pairs, 20K knowledge question-answering pairs, and 200+ skill-code pairs. We conduct continuous block search, knowledge question and answering, and tech tree mastery to evaluate the performance. Extensive experiments show that STEVE achieves at most 1.5× faster unlocking key tech trees and 2.5× quicker in block search tasks.

Keywords: Open-world embodied agent · Multimodal pre-training · Large language model

1 Introduction

Designing agents that demonstrate intelligent behavior and adaptability in open-world settings has been a longstanding and significant challenge in the field of artificial intelligence [13,14,26,46,70]. However, recent progress in the development of large language models (LLMs) [5, 55] has exhibited their potential as versatile, general-purpose assistants. Recent innovations in agent design [57,59,69,77]

♠ Equal contribution: zhaozhonghan@zju.edu.cn, wchai@uw.edu, xuanw@zju.edu.cn.
† Corresponding author: gaoangwang@intl.zju.edu.cn.

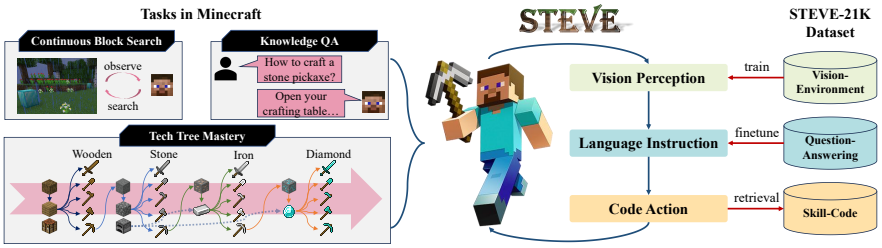

Fig. 1: STEVE integrates the three parts: Vision Perception, Language Instruction, and Code Action, supported closely by our proposed STEVE-21K. It demonstrates commendable performance on Continuous Block Search, Knowledge QA, and Tech Tree Mastery.

have effectively harnessed these advanced LLMs, tapping into their extensive world knowledge and reasoning abilities. This development has paved the way for agents, that are autonomously driven, to formulate and implement strategies and actions across a diverse array of skills and tasks in diverse open-world environments.

In many open-world settings, like Minecraft, contemporary agents predominantly use LLMs for their textual interactions. However, this reliance on text for communication poses considerable limitations in their interactions within these worlds including low-level regrading cases [24, 56, 65, 66]. Minecraft, with its expansive and interactive sandbox environment [17, 20], demands a variety of skills from agents, ranging from crafting basic items to executing complex tasks. Yet, agents driven by LLMs often generate unpredictable outputs. The effectiveness of their interactions is largely contingent on meticulously crafted prompts [23], designed to align the LLM's understanding with the environmental context and the intended objectives. This process of prompt engineering is not only laborious but also fails to meet the goal of fostering autonomous, self-directed agents. Furthermore, textual communication has its limitations in naturally conveying certain concepts of the world, like crafting recipes, which are often more effectively communicated through vision.

Players have the distinct capability to assimilate and convey information using both visual and textual channels, significantly enhancing our interactions with the world around us. Yet, the integration of LLM-based agents with multimodal inputs in open-ended environments remains an under-explored area. STEVE in STEVE-Series [71–73], is named after the protagonist of the game "Minecraft,". It is our proposed framework to build an embodied agent based on the vision model and LLMs within an open world, as illustrated in Fig. 1. STEVE harnesses a vision model to perceive its surroundings visually, coupled with an LLM to strategize and plan actions. This model represents a leap forward in agent design, combining these two input modes, vision, and text, to offer a more nuanced and comprehensive understanding of the environment and practical and executable skills.

Our key contributions are outlined as follows:

- We propose STEVE, an embodied agent in virtual environment, consists of vision perception, language instruction, and code action, achieving 1.5× faster unlocking of key tech trees and is 2.3× quicker in block search tasks compared to previous state-of-the-art methods.
- We present STEVE-7B/13B, a series of large language model obtained by fine-tuning with Minecraft knowledge question-answering pairs from Llama-2-7B/13B.
- We collect STEVE-21K dataset, including 600+ vision-environment pairs, 20K knowledge question-answering pairs, and 200+ skill-code pairs, for justifying the effective performance of STEVE.

2 Related Works

2.1 Intelligent Agent in Minecraft

As an open-ended sandbox game, Minecraft has always been an ideal setting for testing the performance of intelligent agents [21, 25]. The agents are required to autonomously perform various tasks in Minecraft, such as chopping trees, crafting tools, and mining diamonds. At the beginning, much of the works focus on exploring reinforcement learning [32, 33, 42, 50] or imitation learning [2, 3], without satisfactory performance. VPT [3] and MineDojo [17] collect internet-scale datasets for their model pre-training. More specifically, VPT offers the exciting possibility of directly learning to act during video pre-training and using these learned behavioral priors as extremely effective exploration priors for reinforcement learning. Yet, recent works found that the pre-trained LLMs could serve as a strong "mind" that provides planning ability to the agents. Voyager [57] leverages GPT-4 [44] as both a high-level planner and a low-level action code generator. Plan4MC [69] proposes a skill graph pre-generated by the LLMs. DEPS [59], an interactive planning method based on LLMs, addresses multi-step reasoning issue in open-world planning. GITM [77] develops a set of structured actions and leverages LLMs to generate action plans for the agents to execute, achieving impressive results in various tasks.

2.2 Embodied Multimodal Model

Embodied agent operates within various environment by synthesizing sensory perceptions and physical actions supported by computational intelligence. This synthesis enables the agent to undertake a variety of tasks, achieving specific objectives. Its key areas of application are diverse, including Navigation [6, 16, 27, 43, 61, 68], Embodied Question Answering [10, 11, 67], Active Visual Tracking [37, 38, 74, 75], and Visual Exploration [7, 12, 35]. The field is evolving rapidly with the development of Large Language Models (LLMs) [51] and Multimodal LLMs (MLLMs) [1, 9, 18, 19, 28–30, 34, 39, 41, 54, 58, 64, 76], integrating multiple modalities for more effective processing. A prime example of this innovation is

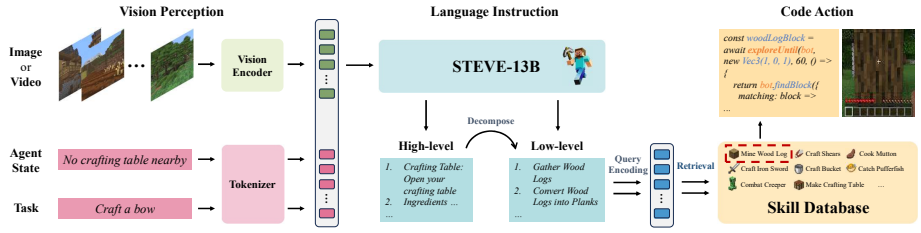

Fig. 2: STEVE framework. The Vision Perception part takes images or videos, encodes them into tokens, and combines them with Agent State and Task tokens as input. The STEVE-13B in the Language Instruction part is used for automatic reasoning and task decomposition, and it calls the Skill Database in the form of the Query to output code as action.

PaLM-E [15], a sophisticated multimodal model with 562B parameters, adept at a broad spectrum of embodied tasks and demonstrating exceptional capabilities in visual reasoning.

2.3 Large Language Model with Equipped Tools

While Large Language Models (LLMs) demonstrate impressive skill in tackling novel tasks via prompt-based instructions, they often face challenges in areas where simpler models or tools excel, like mathematical calculations or identifying palindromes. However, LLMs' potential is significantly expanded when integrated with other modality-specific models, such as those for vision or audio, enabling multi-modal capabilities [4,36]. Innovations like Toolformer [47] demonstrate LLMs' self-learning to utilize tools through finetuning with extensive API call samples. Visual ChatGPT [62] extends this by integrating various visual foundation models, facilitating interactive user experiences with ChatGPT. Similarly, HuggingGPT [48] presents a framework that harnesses LLMs to link diverse models from Hugging Face for task resolution. AutoGPT [49] is an open-source application that broadens GPT-4's capabilities with internet access, memory management, and plug-ins. The recent introduction of MovieChat [52,53] brings a memory mechanism to MLLM, enhancing its performance in video understanding tasks. Furthermore, LLMs can be used for goal planning, analogous to language translation [63]. This evolving landscape suggests that tool-equipped LLMs could forge a new paradigm in AI solution design.

3 Method: STEVE

3.1 Overview

As shown in Figure 2, STEVE $\mathcal{F}$ is an LLM-based multi-modal autonomous system for embodied agents in Minecraft, which can use visual status X^v in the

form of images or videos and agent state X^s to manage and execute complex task X^t for executive code action $\mathbf{a}^c$:

$$\mathbf{a}^c = \mathcal{F}(X^v, X^s, X^t) = \mathcal{A}^c(\mathcal{I}^l(\mathcal{P}^v(X^v, X^s, X^t))) \tag{1}$$

where STEVE $\mathcal{F}$ integrates Visual Perception $\mathcal{P}^v$ into the Language Instruction $\mathcal{I}^l$ with large language model (LLM) and combines them with Code Action $\mathcal{A}^c$, a skill retrieval method to execute actions, details are demonstrated as follows.

- **Vision Perception $\mathcal{P}^v$ (Section 3.2)**; a vision encoder to interpret visual information of the environment, such as blocks and entities and a text tokenizer for agent state s and task t.
- **Language Instruction $\mathcal{I}^l$ (Section 3.3)**: a powerful language model system fine-tuned specifically for Minecraft content using LLaMA2-13B [18]. This model enables adaptive interaction for iterative reasoning and step-by-step decomposition.
- **Code Action $\mathcal{A}^c$ (Section 3.4)**: a skill retrieval method based on our skill-code database.

3.2 Vision Perception $\mathcal{P}^v$

The vision perception part includes a vision encoder E^v and a text tokenizer T, which converts the visual status X^v, agent state X^s, and task X^t to the text-space tokenizer representation $\mathbf{Y} = \{Y^v, Y^s, Y^t\}$:

$$\mathbf{Y}_i = \mathcal{P}^v(X_i^v, X_i^s, X_i^t), \mathcal{P}^v = \{E^v, T\}, \tag{2}$$

where the vision encoder E, the visual branch of EfficientFormer [31], encodes visual status X_i^v at each step i into visual tokens $Y^v = \{y_1^v, y_2^v, ..., y_n^v\} \in \mathbb{R}^{n \times d}$, where n denotes the number of visual tokens y^v and d is the dimensionality of each token. The text tokenizer converts agent state X_i^s and task X_i^t at each step i in the form of text into textual tokens Y^s and Y^t. Note that visual tokens are amalgamated with textual tokens representing the agent's current state (*e.g.*, health, inventory, *etc.*) and the task description. This is accomplished using a tokenizer that maps state variables and task parameters to a tokenized form. The resultant unified token set serves as a comprehensive representation of the current situational context.

3.3 Language Instruction $\mathcal{I}^l$

The Language Instruction $\mathcal{I}^l = \{\mathcal{P}_l, \mathcal{C}_r, \mathcal{C}_u, \mathcal{D}_s\}$, which consists of Planner $\mathcal{P}_l$, Critic $\mathcal{C}_r$, Curriculum $\mathcal{C}_u$, and Describer $\mathcal{D}_s$, *i.e.*, four independent LLM-based agents with different functions. They formulate high-level task guidelines, refine strategies through feedback, facilitate continuous learning and adaptation through a curriculum of complex tasks, and finally decompose strategic guidelines into the executable low-level textual action step $\mathbf{a^s}$, $\{\mathbf{a^s} \sim \mathcal{I}^l(\cdot|\mathbf{Y}_i)\}_{i=0}^N$ towards efficient completion of the task.

Note that the Planner $\mathcal{P}_l$, Critic $\mathcal{C}_r$, Curriculum $\mathcal{C}_u$ and Describer $\mathcal{D}_s$ are based on STEVE-7B/13B, a powerful language model derived from LLaMA-2-13B [18], fine-tuned specifically on Minecraft-related content from the STEVE-21K. This model's expertise covers a broad spectrum of game-specific knowledge areas on worlds, entities, player mechanics, survival, and even game practical experience. Finally, they have the performance of their different functions:

- **Planner $\mathcal{P}_l$:** formulating comprehensive guidelines and executive plans that align with the overarching objectives of the task.
- **Critic $\mathcal{C}_r$:** evaluating the planner decisions, providing feedback that can refine strategies.
- **Curriculum $\mathcal{C}_u$:** facilitating continuous learning and adaptation for action agents by engaging with a series of progressively complex tasks.
- **Describer $\mathcal{D}_s$:** distilling the extensive data into a concise summary, making it more manageable and interpretable.

Iterative reasoning. The STEVE-13B receives a stream of tokens that encode the current visual scene, the agent's state, and the task's textual description. STEVE-13B interprets this rich context to undertake complex reasoning. The model initiates the reasoning process by constructing a series of high-level strategies that outline the pathway to task completion. Considering all gameplay elements, the reasoning mechanism is akin to an experienced player who can visualize the end game and chart a course to victory. This approach ensures the plans are reactive and proactive, allowing the agent to anticipate and mitigate future challenges. However, most strategies are high-level and abstract therefore, they often require step-by-step decomposition to derive executable guidelines.

Decomposition. The decomposition process makes the complex strategies break down into simple, low-level guidelines that can be directly mapped to actions in Minecraft. It is similar to how a high-level command like "build a shelter" is divided into actionable instructions like "collect wood", "craft planks", and "place blocks". The granular steps are structured to provide explicit instructions that the game engine can readily interpret. This systematic breakdown from high-level reasoning to low-level actionable steps is the hallmark of the STEVE system, enabling the embodied agent to interact with the Minecraft environment in a meaningful and goal-oriented manner. Through this intricate process of reasoning and decomposition, STEVE embodies the cognitive capabilities required for sophisticated task management in virtual environments.

Curriculum learning with memory. We draw inspiration from the lifelong learning strategy utilized in many reinforcement learning problems [57] in both closed-world and open-world settings. We start by creating a set of tasks that serve as a curriculum for agents to explore the environment. During this process, STEVE generates plans, interacts with the surroundings, learns from mistakes, and stores all these experiences in memory. Next, we evaluate STEVE on various tasks after this learning stage. Consequently, STEVE can produce better plans with its

memory teaming up with the planning experiences. We use this as the default setting for all tasks in our experiments.

Continous learning with summarization. We've observed that the learning process, where the memory is being filled, can continue throughout the gameplay. The agent can gradually acquire more skills as the gameplay progresses and more experiences are gained. However, as the memory gets larger, it becomes difficult to understand the game's situation and interact with the game slowly. To tackle these challenges, we have implemented the Chain of Summarization method [40]. By finding better references, we can improve our ability to handle challenging tasks, such as "Diamond Tool" in the tech tree, including obtaining materials and creating diamond tools. This will lead to a higher success rate, as shown in Section 5.3. Additionally, curriculum learning with memory allows for in-context lifelong learning without needing gradient updates.

3.4 Code Action $\mathcal{A}^c$

The code action part $\mathcal{A}^c$ is the execution phase, where the STEVE system converts planned, decomposed guidelines into concrete actions within the Minecraft environment. This process leverages a specialized skill database that pairs code snippets with their descriptions and relevant metadata, encoded as vectors $\mathbf{v^s}$ for efficient retrieval. The transition from language instruction to executable code is achieved through the retrieval process $\mathcal{R}$:

$$\mathcal{R}(\mathbf{q}, \mathbf{v}) = \sigma(\mathbf{q}, \mathbf{v}), \mathbf{q} = E^q(\mathbf{a^s}), \tag{3}$$

where E^q and σ are query encoding and cosine similarity matching. Each low-level textual action step $\mathbf{a^s}$ derived from the Language Instruction is encoded into a query $\mathbf{q}$. This encoding captures the essence of the action to be performed in a format that the skill database can understand. Once the queries $\mathbf{q}$ are encoded, the system computes similarities between the query vectors and the code snippets vectors $\mathbf{v}$ stored in the database to determine which skill best matches the required action.

3.5 Training

To reduce the training overhead to a certain extent, we adopt a two-stage training method, the first stage being a warm-up on the question-answering pairs of STEVE-21K to ensure a certain degree of instruction capability. The second stage is simulated in the environment, and the Expert LLM $\mathcal{E}_p$ integrated with GPT-4 [44] generates instructions for the same situation to modify our model.

In training, we use either a single-round conversation $\{X^Q, X^A\}$ or a multi-round conversation $\{X_i^Q, X_i^A\}_{i \leq N}$, where N is the total number of rounds in the conversation. We use the negative log-likelihood objective over the prediction

tokens with training and finetuning:

$$\mathcal{L}(\theta) = -\sum_{j=1}^{L} \log \mathcal{F}_\theta(Y_j | X^v, \hat{Y}_{1:j-1}), \tag{4}$$

where Y and $\hat{Y}$ refer to the non-vision input and target token sequences, θ represents the model parameters, and L represents the length of the target sequence. The vision input X^v varies depending on the input step. We only consider the answer tokens X_A when computing the loss to ensure that the model focuses on generating coherent responses.

Offline warm-up. In the first stage, we finetune the STEVE-7B/13B only on the single-round question-answering pairs of STEVE-21K to ensure a certain degree of instruction capability, as shown in Tab. 3.

Online finetuning. In the second stage, we simulate the warmed-up STEVE-7B/13B in Minecraft, and both train the vision encoder E^v and finetune the STEVE-7B/13B simultaneously.

It is necessary to obtain the environment information to train the vision encoder, denoted as $l = R_T(X^v)$, where l is the label of seen blocks and entities, R_T is the Ray Tracing method [60] to eliminate all content outside the agent's perspective screen. We train the vision encoder E^v once we have obtained the environment information. To finetune the warmed-up STEVE-7B/13B in simulation, we give both the STEVE-7B/13B and the Expert LLM the same input, and we consider the output of the Expert LLM as the ground truth label.

After 5,000 simulations, we collect all the context information from successful runs, including sequences of vision and question-answering chat flow as the Vision-Environment part of STEVE-21K, as mentioned in Section 4.

4 Dataset: STEVE-21K

As shown in Fig. 3, STEVE-21K has been meticulously compiled, featuring a diverse collection of Vision-Environment pairs, Question-Answering pairs, and a Skill-Code database.

Vision-Environment pairs contain 600 pairs of first-person perspective videos from Minecraft gameplay across six different terrains (including forest, desert, coastal, *etc.*), along with corresponding environmental block entities in the field of vision and context information for each timestamp. Additionally, all pairs are oriented around executing the skill-related task supported by Skill-Code part mentioned in Section 4. We employ the STEVE-7B/13B model to enable robots to plan and execute actions autonomously based on tasks defined by human supervisors. We record the video of the agent operation, the environment information, and all the corresponding chatflow.

Question-Answering pairs contain 20K question-answering pairs from the Minecraft-Wiki and Reddit corpus across six data types partly sourced from [17]. The pairs are organized into instruction, input, and output triplets and used to

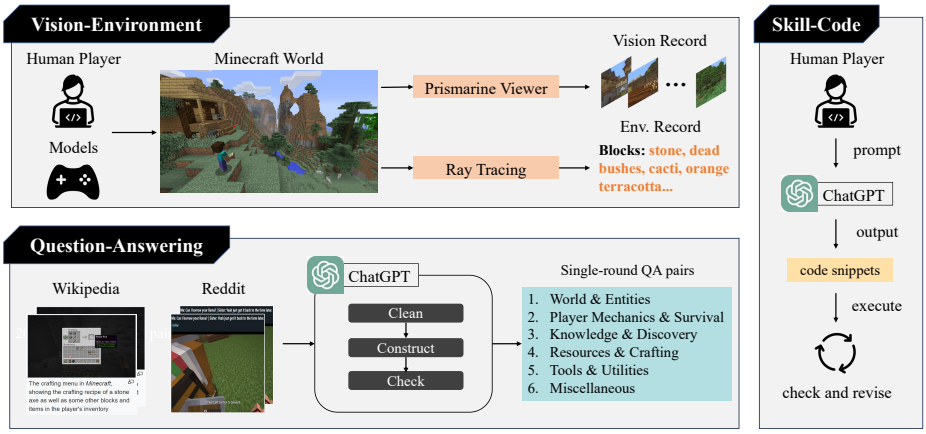

Fig. 3: STEVE-21K collection pipeline. In the Vision-Environment section, STEVE-13B plays the game according to specified tasks defined by the human player, collecting visual information through prismarine-viewer and capturing environmental information from the screen using Ray Tracing [60]. Note that the language instruction task is also performed during the collection phase. We simultaneously record and save the chat flow from the reasoning and decomposition stages. In the Question-Answering section, we obtain information from the Minecraft-Wiki and Reddit forums and use GPT-3.5 to clean the data into single-round QA pairs. In the Skill-Code section, we use GPT-3.5 combined with the human player's code to synthesize code snippets and then check and revise them in the game environment.

train STEVE-13B. The GPT-3.5 [5] is employed to derive meaningful single-round question-answer pairs, and LoRA [22] is incorporated during the fine-tuning process for efficient resource allocation.

Skill-Code pairs contain 210 skill execution scripts with descriptions, covering 8 skill types including collecting, crafting, exploration *etc*. The code part is collected by manual coding. We use GPT-3.5 [5] to describe all codes and utilize langchain vectordb to give all pairs a database vector.

5 Experiments

5.1 Experimental Setup

We train STEVE-7B/13B, which is finetuned from LLaMA-2 [18] with the Queston-Answering pair in STEVE-21K dataset for warm-up and simulation context data from successful runs. We use LoRA [22] for finetuning process. Note that the process is to adjust STEVE-13B to work on correct simulation knowledge while remaining adapted to visual perception. In the text part, we set all temperatures to 0 except for the task proposal, which uses 0.9 to encourage task diversity. The vision unit is based on EfficientFormerV2-S0 [31], which is trained on the Vision-Environment part of our STEVE-21K dataset. Our simulation environment is built on top of MineDojo [17] and leverages Mineflayer [45].

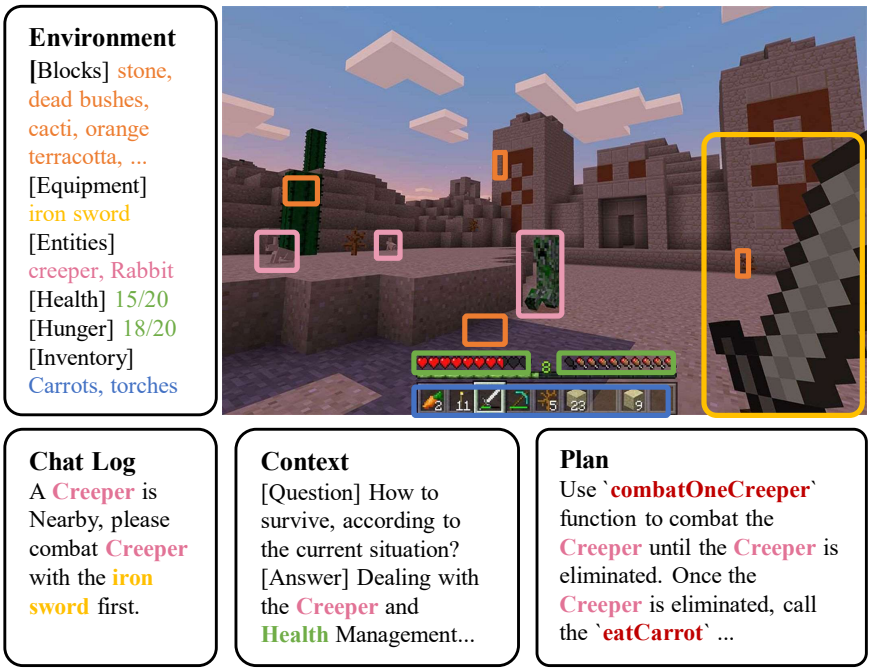

Fig. 4: Example of Vision-Environment pairs. It represents the data format of the Vision-Environment pairs in our STEVE-21K dataset: including visual signals, environmental information, Chat Log, Context QA pairs, and planning in actual tasks.

We use GPT-4-0613 for all GPT-4 models used in Voyager [57] and code generation tasks.

5.2 Baselines

As no vision-based LLM-driven agents are immediately operable in Minecraft, we selected several algorithms as baselines that extract information from a system's backend, differing significantly from real-world applications.

AutoGPT [49] is an NLP automation tool that decomposes a high-level goal into executable subgoals in MineDojo, aligning with our experimental framework. Our setup, AutoGPT, powered by GPT-4 [44], processes agent states, environment feedback, and execution errors to manage subgoal execution.

Voyager [57] relies solely on textual grounding for perception and features a long-term procedural memory with a hierarchical library of code-based procedures, allowing the integration of simple skills into complex behaviors. Proficient in environment exploration and tech tree mastery, Voyager uses GPT-4 [44] for processing background text in embodied agents, with a lesser emphasis on visual perception.

Table 2: Comparison on **tech tree mastery** task. The values presented are in fractions, representing successful trials out of three attempts. A score of 0/3 signifies the method's inability to progress within the tech tree after a maximum of 160 prompting iterations. The reported numbers denote the average iterations across three trials. Lower iteration values indicate higher efficiency of the respective method.

Method	Wooden Tool	Stone Tool	Iron Tool	Diamond Tool
AutoGPT [49]	92 ± 72 (3/3)	94 ± 72 (3/3)	135 ± 103 (3/3)	N/A (0/3)
Voyager [57]	6 ± 2 (3/3)	11 ± 2 (3/3)	21 ± 7 (3/3)	**102** (1/3)
STEVE	$\mathbf{4 \pm 1}$ (3/3)	$\mathbf{8 \pm 1}$ (3/3)	$\mathbf{15 \pm 2}$ (3/3)	106 ± 12 (3/3)

Table 3: Quantitive comparison on knowledge question and answering task. Questions, model-generated responses, and ground truth inputs are evaluated in GPT-4 [5], Claude-2 [8] and human blind rating rated on a scale of 0 to 10; The scores above are the average of them. Higher scores indicate greater alignment of the generated answers with the ground truth. Wld., Ent., Mech., Surv., Know., Disc., Res., Craft., Tools, Util., Miscell. stand for World, Entities, Player Mechanics, Survival, Knowledge, Discovery, Resources, Crafting, Tools, Utilities and Miscellaneous.

Method	Wld. & Ent.	Mech. & Surv.	Know. & Disc.	Res. & Craft.	Tl. & Util.	Miscell.	Overall
Llama2-7B	6.44	6.68	6.58	6.42	6.80	6.96	6.56
Llama2-13B	6.93	6.95	6.77	6.77	6.98	6.64	6.89
STEVE-7B	7.99	7.88	7.84	7.95	7.93	7.82	7.94
STEVE-13B	**8.14**	**8.13**	8.03	**8.15**	**8.12**	7.72	**8.12**
GPT-4	8.06	8.07	**8.07**	7.92	8.09	**8.21**	8.04

5.3 Evaluation Results

Continuous block search. As shown in Tab. 1, we experiment with block-searching tasks to assess the agent's exploratory capabilities and proficiency in locating specified blocks. Diamond blocks are placed at every 16-block interval across the land map. The agent's objective is to identify as many blocks as possible within the fewest iterations, which indicates the method's efficiency. As shown in Fig. 5, enriching information through visual perception significantly enhances the efficiency of search and exploration tasks, leading to more effective world exploration.

Table 1: Comparison on continues block search task. # Iters stand for average iterations to find 10 diamonds (max 100). # Blocks stand for average diamonds found in 100 iterations.

Method	# Iters (↓)	# Blocks (↑)
AutoGPT [49]	N/A	7
Voyager [57]	35	26
STEVE	**14**	**67**

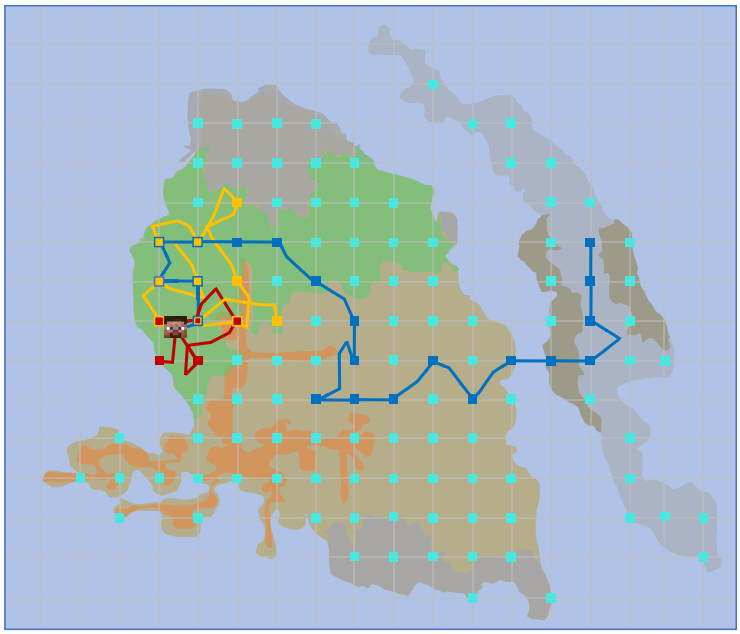

Fig. 5: Schematic of the continuous block search task. We capture an asynchronous segment with each method 30 iterations from the experiments for illustration. The reason we choose diamond blocks is that they are not naturally occurring in the given context, making them easily distinguishable from other blocks.

Knowledge question and answering. Using a validation dataset, we established a question-answering database to evaluate our model's performance on Minecraft-related queries. Responses from each model are rated blindly by GPT-4, Claude-2, and human participants based on accuracy, relevance, and detail. Responses are initially checked for accuracy, and the comprehensive evaluation yields a score ranging from 0 to 10, with higher scores indicating superior performance.

As detailed in Tab. 3, we tested the instructional capabilities of various LLM models using the STEVE-21k dataset, divided into $18,622$ training and $1,000$ testing samples. The STEVE-7B and STEVE-13B models surpassed LLaMA2 in all metrics, with STEVE-13B scoring the highest at 8.12, demonstrating its exceptional understanding of Minecraft-related queries. This indicates that STEVE models, optimized specifically for Minecraft content, perform better in knowledge-intensive tasks.

Our results confirm that larger, domain-specifically tuned models like STEVE-13B outperform broader models like GPT-4, underscoring the benefit of specialized fine-tuning for domain-specific applications.

Fig. 6: Qualitative comparison on knowledge question and answering tasks. Green (Red) highlights the correct or good (wrong or bad) answer. Blue indicates the suboptimal answer. Grey indicates the meaningless answer.

Tech tree mastery. As shown in Tab. 2, we experiment on the Minecraft tech tree mastery to test the agent's ability to craft and use a hierarchy of tools. Progressing through this tree (wooden tool → stone tool → iron tool → diamond tool) requires the agent to master systematic and compositional skills. As to the wooden, stone, and iron levels of the tech tree, STEVE achieves remarkable efficiency: 23×, 11.8×, and 9× faster than AutoGPT [49], and 1.5×, 1.4×, and 1.3× faster than Voyager [57]. STEVE has achieved the diamond level, as shown in Tab. 2. Its performance slightly lags behind Voyager [57], which also uses GPT4 [44] for critical inference. However, STEVE is more cost-effective, starting with lower initial performance. It includes a vision unit, prioritizing visual data over background information, offering distinct advantages. Additionally, we observed a decrease in performance when using a basic skill database.

5.4 Ablation Study

To understand the impact of different components on the performance of our system, we conducted ablation studies focusing on the tech tree mastery task in Minecraft. The results, as shown in Tab. 4, provide insights into the effectiveness of the vision unit and compare our STEVE model with the STEVE GPT-4 version (with the same vision unit as ours). Note that the w/o vision unit setup is that the environmental perception encompasses data on blocks within an 8x8 area surrounding the agent, including the front, back, left, and right directions. The following observations are made:

Table 4: Ablation studies for the tech tree mastery. STEVE (Ours) is the STEVE-13B version. The 0/3 score means the method can't progress beyond 160 iterations in the tech tree.

Method	Wooden Tool	Stone Tool	Iron Tool	Diamond Tool
w/o vision unit	11 ± 5 (3/3)	27 ± 5 (3/3)	46 ± 11 (3/3)	158 (1/3)
STEVE (GPT-4)	6 ± 2 (3/3)	10 ± 1 (3/3)	**14 ± 3 (3/3)**	**89 ± 9 (3/3)**
STEVE (Ours)	**4 ± 1 (3/3)**	**8 ± 1 (3/3)**	15 ± 2 (3/3)	106 ± 12 (3/3)

Vision unit is critical. The omission of the vision unit markedly affects the system's performance, especially in more advanced tasks. While it successfully crafts Wooden, Stone, and Iron Tools, it is challenged with Diamond Tools. This outcome underscores the vital importance of visual information in accomplishing complex tasks.

Comparison with GPT-4. As our vision encoder directly encodes into text space, it can be easily replaced with any language model. For instance, the GPT-4 we compared exhibits consistent success across all categories and secures a flawless success rate. Interestingly, the STEVE-13B version excels in simpler tasks such as crafting wooden and stone tools. Moreover, it requires fewer iterations than methods without the vision part, underscoring its superior efficiency.

5.5 Case Study

As shown in Fig. 6, we perform an extensive case study comparison of GPT-4, LLaMA2-13B, and our method STEVE-13B. Each model maintains the same information and question inputs to compare feedback under different environmental information. Our STEVE overall achieves the best results, surpassing GPT-4 and showing significant improvement compared to the original LLaMA. Especially in parts involving numerical calculations, such as the leftmost image, STEVE accurately tracks food values to restore hunger levels.

6 Conclusion

STEVE enhances multi-modal learning by combining visual encoder and LLM-based agents. It has three functions: vision perception, language instruction, and code action, allowing it to understand, predict, and act in virtual environments. We provide a straightforward approach to creating a robust, multi-modal, autonomous, embodied agent using an open-source language model with a small number of parameters. Additionally, we provide a comprehensive dataset STEVE-21K for sustainable community development that can be verified.

Acknowledgement

This work is supported by the Zhejiang Provincial Natural Science Foundation of China (No. LZ24F030005) and the National Natural Science Foundation of China (No. 62106219).

References

1. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a visual language model for few-shot learning. Advances in Neural Information Processing Systems **35**, 23716–23736 (2022)
2. Amiranashvili, A., Dorka, N., Burgard, W., Koltun, V., Brox, T.: Scaling imitation learning in minecraft. arXiv preprint arXiv:2007.02701 (2020)
3. Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., Clune, J.: Video pretraining (vpt): Learning to act by watching unlabeled online videos. Advances in Neural Information Processing Systems **35**, 24639–24654 (2022)
4. Chai, W., Wang, G.: Deep vision multimodal learning: Methodology, benchmark, and trend. Applied Sciences **12**(13), 6588 (2022)
5. Introducing chatgpt (2022), `https://openai.com/blog/chatgpt`
6. Chen, S., Guhur, P.L., Schmid, C., Laptev, I.: History aware multimodal transformer for vision-and-language navigation. Advances in neural information processing systems **34**, 5834–5847 (2021)
7. Chen, T., Gupta, S., Gupta, A.: Learning exploration policies for navigation. In: International Conference on Learning Representations (2018)
8. Talk to claude (2023), `https://claude.ai`
9. Dai, W., Li, J., Li, D., Tiong, A.M.H., Zhao, J., Wang, W., Li, B., Fung, P., Hoi, S.: Instructblip: Towards general-purpose vision-language models with instruction tuning. arXiv preprint arXiv:2305.06500 (2023)
10. Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., Batra, D.: Embodied question answering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–10 (2018)
11. Datta, S., Dharur, S., Cartillier, V., Desai, R., Khanna, M., Batra, D., Parikh, D.: Episodic memory question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19119–19128 (2022)
12. Dean, V., Tulsiani, S., Gupta, A.: See, hear, explore: Curiosity via audio-visual association. Advances in neural information processing systems **33**, 14961–14972 (2020)
13. Deng, J., Chai, W., Guo, J., Huang, Q., Hu, W., Hwang, J.N., Wang, G.: Citygen: Infinite and controllable 3d city layout generation. arXiv preprint arXiv:2312.01508 (2023)
14. Deng, J., Chai, W., Huang, J., Zhao, Z., Huang, Q., Gao, M., Guo, J., Hao, S., Hu, W., Hwang, J.N., et al.: Citycraft: A real crafter for 3d city generation. arXiv preprint arXiv:2406.04983 (2024)
15. Driess, D., Xia, F., Sajjadi, M.S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al.: Palm-e: An embodied multimodal language model. arXiv preprint arXiv:2303.03378 (2023)

16. Du, H., Yu, X., Zheng, L.: Vtnet: Visual transformer network for object goal navigation. In: International Conference on Learning Representations (2020)
17. Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.A., Zhu, Y., Anandkumar, A.: Minedojo: Building open-ended embodied agents with internet-scale knowledge. Advances in Neural Information Processing Systems **35**, 18343–18362 (2022)
18. Gao, P., Han, J., Zhang, R., Lin, Z., Geng, S., Zhou, A., Zhang, W., Lu, P., He, C., Yue, X., et al.: Llama-adapter v2: Parameter-efficient visual instruction model. arXiv preprint arXiv:2304.15010 (2023)
19. Gong, T., Lyu, C., Zhang, S., Wang, Y., Zheng, M., Zhao, Q., Liu, K., Zhang, W., Luo, P., Chen, K.: Multimodal-gpt: A vision and language model for dialogue with humans. arXiv preprint arXiv:2305.04790 (2023)
20. Guss, W.H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., Salakhutdinov, R.: Minerl: A large-scale dataset of minecraft demonstrations. arXiv preprint arXiv:1907.13440 (2019)
21. Hofmann, K.: Minecraft as ai playground and laboratory. In: Proceedings of the annual symposium on computer-human interaction in play. pp. 1–1 (2019)
22. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
23. Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al.: Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608 (2022)
24. Jiang, J., Ye, T., Bai, J., Chen, S., Chai, W., Jun, S., Liu, Y., Chen, E.: Five a+ network: You only need 9k parameters for underwater image enhancement. arXiv preprint arXiv:2305.08824 (2023)
25. Johnson, M., Hofmann, K., Hutton, T., Bignell, D.: The malmo platform for artificial intelligence experimentation. In: Ijcai. pp. 4246–4247 (2016)
26. Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Deitke, M., Ehsani, K., Gordon, D., Zhu, Y., et al.: Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474 (2017)
27. Kwon, O., Park, J., Oh, S.: Renderable neural radiance map for visual navigation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9099–9108 (2023)
28. Li, B., Zhang, Y., Chen, L., Wang, J., Yang, J., Liu, Z.: Otter: A multi-modal model with in-context instruction tuning. arXiv preprint arXiv:2305.03726 (2023)
29. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv preprint arXiv:2301.12597 (2023)
30. Li, J., Li, D., Xiong, C., Hoi, S.: Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: International Conference on Machine Learning. pp. 12888–12900. PMLR (2022)
31. Li, Y., Yuan, G., Wen, Y., Hu, J., Evangelidis, G., Tulyakov, S., Wang, Y., Ren, J.: Efficientformer: Vision transformers at mobilenet speed. Advances in Neural Information Processing Systems **35**, 12934–12949 (2022)
32. Lifshitz, S., Paster, K., Chan, H., Ba, J., McIlraith, S.: Steve-1: A generative model for text-to-behavior in minecraft (abridged version). In: NeurIPS 2023 Workshop on Goal-Conditioned Reinforcement Learning (2023)
33. Lin, Z., Li, J., Shi, J., Ye, D., Fu, Q., Yang, W.: Juewu-mc: Playing minecraft with sample-efficient hierarchical reinforcement learning. arXiv preprint arXiv:2112.04907 (2021)

34. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. arXiv preprint arXiv:2304.08485 (2023)
35. Liu, S., Okatani, T.: Symmetry-aware neural architecture for embodied visual exploration. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17221–17230. IEEE (2022)
36. Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.W., Wu, Y.N., Zhu, S.C., Gao, J.: Chameleon: Plug-and-play compositional reasoning with large language models. arXiv preprint arXiv:2304.09842 (2023)
37. Luo, W., Sun, P., Zhong, F., Liu, W., Zhang, T., Wang, Y.: End-to-end active object tracking via reinforcement learning. In: International conference on machine learning. pp. 3286–3295. PMLR (2018)
38. Luo, W., Sun, P., Zhong, F., Liu, W., Zhang, T., Wang, Y.: End-to-end active object tracking and its real-world deployment via reinforcement learning. IEEE transactions on pattern analysis and machine intelligence **42**(6), 1317–1332 (2019)
39. Lyu, C., Wu, M., Wang, L., Huang, X., Liu, B., Du, Z., Shi, S., Tu, Z.: Macaw-llm: Multi-modal language modeling with image, audio, video, and text integration. arXiv preprint arXiv:2306.09093 (2023)
40. Ma, W., Mi, Q., Yan, X., Wu, Y., Lin, R., Zhang, H., Wang, J.: Large language models play starcraft ii: Benchmarks and a chain of summarization approach. arXiv preprint arXiv:2312.11865 (2023)
41. Maaz, M., Rasheed, H., Khan, S., Khan, F.S.: Video-chatgpt: Towards detailed video understanding via large vision and language models. arXiv preprint arXiv:2306.05424 (2023)
42. Mao, H., Wang, C., Hao, X., Mao, Y., Lu, Y., Wu, C., Hao, J., Li, D., Tang, P.: Seihai: A sample-efficient hierarchical ai for the minerl competition. In: Distributed Artificial Intelligence: Third International Conference, DAI 2021, Shanghai, China, December 17–18, 2021, Proceedings 3. pp. 38–51. Springer (2022)
43. Moudgil, A., Majumdar, A., Agrawal, H., Lee, S., Batra, D.: Soat: A scene-and object-aware transformer for vision-and-language navigation. Advances in Neural Information Processing Systems **34**, 7357–7367 (2021)
44. OpenAI: Gpt-4 technical report. arXiv preprint arXiv: Arxiv-2303.08774 (2023)
45. PrismarineJS: Prismarinejs/mineflayer: Create minecraft bots with a powerful, stable, and high level javascript api. (2013), `https://github.com/PrismarineJS/mineflayer/tree/master`
46. Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al.: Habitat: A platform for embodied ai research. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9339–9347 (2019)
47. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language models can teach themselves to use tools. arXiv preprint arXiv:2302.04761 (2023)
48. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. arXiv preprint arXiv:2303.17580 (2023)
49. Significant-Gravitas: Auto-gpt. `https://github.com/Significant-Gravitas/Auto-GPT` (2023)
50. Skrynnik, A., Staroverov, A., Aitygulov, E., Aksenov, K., Davydov, V., Panov, A.I.: Hierarchical deep q-network from imperfect demonstrations in minecraft. Cognitive Systems Research **65**, 74–78 (2021)

51. Song, C.H., Wu, J., Washington, C., Sadler, B.M., Chao, W.L., Su, Y.: Llm-planner: Few-shot grounded planning for embodied agents with large language models. arXiv preprint arXiv:2212.04088 (2022)
52. Song, E., Chai, W., Wang, G., Zhang, Y., Zhou, H., Wu, F., Guo, X., Ye, T., Lu, Y., Hwang, J.N., et al.: Moviechat: From dense token to sparse memory for long video understanding. arXiv preprint arXiv:2307.16449 (2023)
53. Song, E., Chai, W., Ye, T., Hwang, J.N., Li, X., Wang, G.: Moviechat+: Question-aware sparse memory for long video question answering. arXiv preprint arXiv:2404.17176 (2024)
54. Su, Y., Lan, T., Li, H., Xu, J., Wang, Y., Cai, D.: Pandagpt: One model to instruction-follow them all. arXiv preprint arXiv:2305.16355 (2023)
55. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
56. Vasluianu, F.A., Seizinger, T., Zhou, Z., Wu, Z., Chen, C., Timofte, R., Dong, W., Zhou, H., Tian, Y., Chen, J., et al.: Ntire 2024 image shadow removal challenge report. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6547–6570 (2024)
57. Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A.: Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291 (2023)
58. Wang, W., Chen, Z., Chen, X., Wu, J., Zhu, X., Zeng, G., Luo, P., Lu, T., Zhou, J., Qiao, Y., et al.: Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. arXiv preprint arXiv:2305.11175 (2023)
59. Wang, Z., Cai, S., Liu, A., Ma, X., Liang, Y.: Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. arXiv preprint arXiv:2302.01560 (2023)
60. Whitted, T.: An improved illumination model for shaded display. In: ACM Siggraph 2005 Courses, pp. 4–es (2005)
61. Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., Batra, D.: Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In: International Conference on Learning Representations (2019)
62. Wu, C., Yin, S., Qi, W., Wang, X., Tang, Z., Duan, N.: Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671 (2023)
63. Xie, Y., Yu, C., Zhu, T., Bai, J., Gong, Z., Soh, H.: Translating natural language to planning goals with large-language models. arXiv preprint arXiv:2302.05128 (2023)
64. Ye, Q., Xu, H., Xu, G., Ye, J., Yan, M., Zhou, Y., Wang, J., Hu, A., Shi, P., Shi, Y., et al.: mplug-owl: Modularization empowers large language models with multimodality. arXiv preprint arXiv:2304.14178 (2023)
65. Ye, T., Chen, S., Liu, Y., Chai, W., Bai, J., Zou, W., Zhang, Y., Jiang, M., Chen, E., Xue, C.: Sequential affinity learning for video restoration. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 4147–4156 (2023)
66. Ye, T., Zhang, Y., Jiang, M., Chen, L., Liu, Y., Chen, S., Chen, E.: Perceiving and modeling density for image dehazing. In: European conference on computer vision. pp. 130–145. Springer (2022)
67. Yu, L., Chen, X., Gkioxari, G., Bansal, M., Berg, T.L., Batra, D.: Multi-target embodied question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6309–6318 (2019)
68. Yu, Y., Huang, W., Sun, F., Chen, C., Wang, Y., Liu, X.: Sound adversarial audio-visual navigation. In: International Conference on Learning Representations (2021)

69. Yuan, H., Zhang, C., Wang, H., Xie, F., Cai, P., Dong, H., Lu, Z.: Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. arXiv preprint arXiv:2303.16563 (2023)
70. Zhao, Z., Chai, W., Hao, S., Hu, W., Wang, G., Cao, S., Song, M., Hwang, J.N., Wang, G.: A survey of deep learning in sports applications: Perception, comprehension, and decision. arXiv preprint arXiv:2307.03353 (2023)
71. Zhao, Z., Chai, W., Wang, X., Ma, K., Chen, K., Guo, D., Ye, T., Zhang, Y., Wang, H., Wang, G.: Steve series: Step-by-step construction of agent systems in minecraft. arXiv preprint arXiv:2406.11247 (2024)
72. Zhao, Z., Chen, K., Guo, D., Chai, W., Ye, T., Zhang, Y., Wang, G.: Hierarchical auto-organizing system for open-ended multi-agent navigation. arXiv preprint arXiv:2403.08282 (2024)
73. Zhao, Z., Ma, K., Chai, W., Wang, X., Chen, K., Guo, D., Zhang, Y., Wang, H., Wang, G.: Do we really need a complex agent system? distill embodied agent into a single model. arXiv preprint arXiv:2404.04619 (2024)
74. Zhong, F., Sun, P., Luo, W., Yan, T., Wang, Y.: Ad-vat+: An asymmetric dueling mechanism for learning and understanding visual active tracking. IEEE transactions on pattern analysis and machine intelligence **43**(5), 1467–1482 (2019)
75. Zhong, F., Sun, P., Luo, W., Yan, T., Wang, Y.: Towards distraction-robust active visual tracking. In: International Conference on Machine Learning. pp. 12782–12792. PMLR (2021)
76. Zhu, D., Chen, J., Shen, X., Li, X., Elhoseiny, M.: Minigpt-4: Enhancing vision-language understanding with advanced large language models. arXiv preprint arXiv:2304.10592 (2023)
77. Zhu, X., Chen, Y., Tian, H., Tao, C., Su, W., Yang, C., Huang, G., Li, B., Lu, L., Wang, X., 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)