Execute Long Horizon Action through Prompting Pre-trained Agent in Minecraft

Yuyang Tang
Department of Computer Science
University of Toronto
yy.tang@mail.utoronto.ca

Shijia Liu
Department of Computer Science
University of Toronto
shijia.liu@mail.utoronto.ca

Zexin Xue
Department of Computer Science
University of Toronto
zexin.xue@mail.utoronto.ca

Abstract

In this paper, we expand on the previous research on autonomous agents in Minecraft, specifically STEVE-1. We present a new approach that leverages the reasoning capability of LLM to guide STEVE-1 agents through instructions, allowing them to accomplish long-horizon and multi-step tasks. We view the LLMs as high-level policies, which will implicitly create an action trajectory to complete the specified long-term task. We apply a benchmark with three long-horizon tasks for the agent to complete. Afterward, we analyze the gameplay recordings of the agents and assess the feasibility of our approach. Finally, we analyze the effectiveness of our method and suggest areas for improvement in future studies.

1 Introduction

Recently, there has been great interest in building autonomous agents to handle intricate objectives in open-world environments. Minecraft is an interactive open-world environment with a state and action space closely mimicking the real world, making it a perfect simulation to develop and test autonomous agent systems. However, even in Minecraft, curating an effective agent is difficult due to the dynamic environment, state space, and action space. Furthermore, many objectives are long-horizon tasks composed of several intermediate sub-tasks. For example, the agent has to create the appropriate tools before extracting rare resources. For these reasons, it is impossible to model rewards and policy using conventional reinforcement learning methods, such as Reward Machines. Previously, MineCLIP [4] and STEVE-1 [6] produced policy models conditioned on text instructions and trained using videos of human gameplay, resulting in an agent that can reliably complete immediate actions but not long-horizon tasks.

For this project, we extend the capability of STEVE-1 by using a Large Language Model (LLM) with visual-language capability as a high-level policy. We prompt the LLM with a long-horizon objective for an immediate sub-task executable by the STEVE-1 agent at any given state. This implies the LLM will create a plan with a sequence of short-horizon actions that eventually achieve the specified objective. In addition, when the agent fails to complete a sub-task given by the LLM, the agent will "ask for help" by querying for a new sub-task with an observation of its current surroundings.

We evaluate our method by running several resource collection benchmark tasks with varying degrees of complexity, ranging from "collect dirt" to "collect iron": the first task doesn't require any tool crafting, whereas the latter requires multiple levels of tool crafting and resource collection. We find

Preprint. Under review.

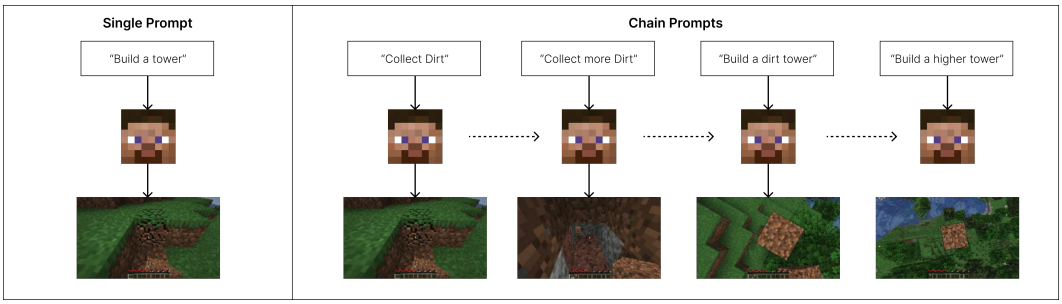

Figure 1: Example of a single prompt and chain prompts. Single prompting (left) is unachievable due to the lack of specification on intermediate steps, such as obtaining the resources to build a tower. Chain prompting (right) specifies those intermediate steps and produces successful results.

STEVE-1 combined with an LLM through curated prompts can usually accomplish complex long-horizon tasks. During our evaluation, we noticed that STEVE-1's low-level policy can hallucinate and perform the wrong action, resulting in the agent's failure. We believe action hallucination emerges from the limited training data used to train STEVE-1. Finally, we identify several areas for improvement and propose plans for future work to improve STEVE-1 and our framework.

2 Related Work

2.1 Planning for Autonomous Agents

The Reward Machine (RM) can define intricate reward functions of an environment. It has been proven to be an effective method in combination with Q-learning for creating an optimal policy [5]. RM is also equivalent to other formal languages that have a temporal dimension, such as Linear Temporal Logic (LTL). By translating complex reward specifications into an automata-like structure, RM can learn the reward value through reward shaping under Q-learning [3]. We hope a formal definition, like RM, can easily break down complex long-horizon objectives into achievable short-horizon actions that STEVE-1 could perform within one prompt. However, while the reward can be learned through Q-learning with reward shaping, the RM structure must be specified for each task. We realized this method is time-consuming and inefficient, especially when dealing with the dynamic and intricate nature of Minecraft.

We drew inspiration from the work of [8] which proposed using LLMs to generate and refine trajectories by asking the LLM to explain its success and failure, and then improve the plan based on the provided explanation. However, in practice, we found this approach to be very cost-efficient as it requires multiple iterations of prompting. Therefore, we decided to directly prompt ChatGPT for refinement without asking for an explanation.

2.2 Large Language Model enabled Autonomous Agents

Large Language Models (LLMs) present some degree of reasoning and planning capability as an emergent ability through their self-supervised training regime. In [7], the researchers connect ChatGPT 3.5 Turbo to autonomous non-playable characters (NPC) in an interactive role-play game environment. Each NPC is an instance of ChatGPT 3.5 Turbo that would keep a memory of the agent's previous states and perform actions accordingly. The agents demonstrated that they can keep track of historic actions and interactions. More importantly, the agents can plan for a long-horizon objective by creating tasks to be completed in sequence over the day, indicating LLMs can function as high-level policies and execute actions to achieve long-horizon objectives.

3 Background: STEVE-1

STEVE-1 [6] is a text-instructable Minecraft agent built on top of two pre-trained models MineCLIP [4] and Video Pre-training (VPT) [1]. The overall framework of STEVE-1 is similar to the unCLIP architecture used for the text-to-image generative model. The MineCLIP model translates the user's

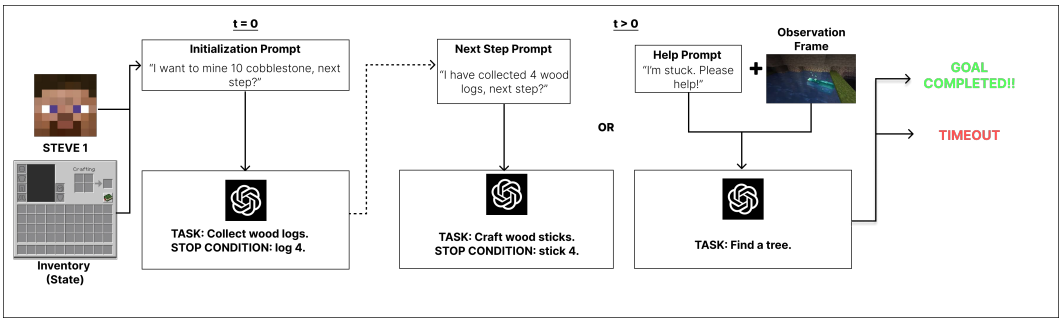

Figure 2: During an episode, STEVE-1 and ChatGPT interact with each other in real-time. At the beginning of an episode, the Initialization prompt is passed into ChatGPT along with the final objective. ChatGPT returns the first action in the trajectory along with the stopping condition of that action. The next prompt is given to STEVE-1 when it completes a task to obtain the following task. If the task is not completed within a given time frame, STEVE-1 sends a help prompt to ChatGPT with the observation frame. This prompt asks ChatGPT to adjust the trajectory of actions accordingly. Finally, the episode ends when the goal is achieved or the specified time limit expires.

text instruction (z_y) to a latent goal embedding (z_{goal}), and it is pre-trained on visually represented goals extracted from online Minecraft gameplay videos [4]. The VPT model combines a new latent goal (z_{goal}), generated by a conditional variational autoencoder (CVAE) from the user instruction (z_y), with control input to map both input types to the corresponding visual goal.

During training, we freeze the weight of MineCLIP and fine-tune the VPT model using a set of randomly selected 16 consecutive gameplay frames paired with their low-level input. This random sequence of frames is called hindsight relabeling. The entire process to fine-tune the VPT is a type of behavior cloning. After fine-tuning the VPT, STEVE-1 can complete actions such as exploring the world and collecting resources with instructions from the user. It is important to note that due to the training regime of MineCLIP and VPT, STEVE-1 requires prompt chaining to complete complicated tasks. We present our version of automated prompt chaining in the next section.

4 Method

4.1 Automated Chain Prompting with LLMs

The results from [6] have shown by adding human-guided chain prompting, STEVE-1 can complete multi-step long-horizon tasks when it cannot with a single prompt. The chain of action prompts can be interpreted as incremental steps building towards the final objective, and each step is a short-horizon task achievable by single prompting STEVE-1 (as shown in Figure 1). We replace the human factor with an LLM to perform high-level reasoning and generate appropriate actions in sequence. Unlike [8] which generates and refines the trajectory through a question-and-answer prompt, our method generates each action on the fly more directly. Moreover, our method refines the trajectory with a new action if the current action is deemed impossible with a specialized help prompt.

Our methodology's workflow is demonstrated in Figure 2. We opted for ChatGPT as the guiding LLM because of its easy-to-use API service and extensive training data, ensuring the model is well-versed in the Minecraft domain. We believe any LLM exposed to Minecraft-related texts during training would also work. Our automated STEVE-1 agent interacts with ChatGPT using three pre-defined prompt templates: Initialization, Next Step, and Help.

To initiate an episode, the agent inputs the specific long-horizon objective and current inventory state into the *Initialization* template and sends this prompt to the ChatGPT instance. The ChatGPT instance responds with the first task and feeds it to STEVE-1 as text instruction. The completion status of this task is then assessed against the stop condition from ChatGPT, which should describe the target inventory state.

After the current task's stop condition is deemed satisfied, the agent invokes the *Next* Prompt, and ChatGPT returns the instruction and stop condition for the next task along the planned trajectory.

This process continues until the STEVE-1 agent reaches the terminal condition. We enable LLM to exert adaptive control on the action trajectory through the *Help* prompt. This approach is equivalent to plan refinement in [8] but is executed dynamically according to the environment.

4.2 Visual Guidance

We anticipate that ChatGPT may encounter difficulties in generating feasible tasks due to a lack of sufficient information about the current local environment. According to the research papers on VPT ([1]), MineCLIP ([4]), and [2], using gameplay visuals can provide an effective representation of an agent's situation. Moreover, ChatGPT-4 can process image inputs with its visual-language capability. Therefore, we propose to present a frame from the agent's observation space, along with the Help prompt, to enable ChatGPT to generate a more optimized trajectory and help the agent overcome the current obstacle (shown in Figure 2).

5 Evaluation

5.1 Benchmark

To evaluate our method's performance on long-horizon objectives, we handcrafted three benchmark tasks that require one or more collection, crafting, and searching actions performed in sequence to complete successfully. All three objectives are resource collection tasks because we can directly evaluate their completion status by observing the agent's inventory state. We understand this short benchmark doesn't capture the diverse objectives in the Minecraft domain, but it is appropriate considering the computation and time constraints. Here are the three benchmark tasks and a sequence of actions to complete that task, which we use as the ground truth:

1. **Collect dirt**: search dirt →dig dirt (2 actions).
2. **Collect wood planks**: search tree →dig wood →craft planks (3 actions).
3. **Collect stone**: collect wood planks →craft pickaxe →search stone →dig stone (with pickaxe) (6 actions).

We perform 3 trials for each benchmark task. Each trial is given 7200 frames (equivalent to 4 minutes of gameplay) in a random seed world. This means the agents will be deployed at a different spawn point and have to devise a unique trajectory that will work for that environment to complete the trial successfully.

For each benchmark task, we perform both quantitative and qualitative analysis. Quantitatively, we measure the success rate within the given step constraint (100,000 steps), efficiency (i.e. amount of time taken to complete the task), and total collected resources. Qualitatively, we record each episode and pinpoint noticeable moments that contribute to the agent's success or failure.

We are limiting the benchmark to three long-horizon tasks because GPT4's vision API enforces a tier system that controls a cap on the number of tokens processable per minute. During the time of our experiment, our newly created accounts are categorized at the lowest tier, resulting in prolonged runtime and experimenting with more long-horizon tasks becoming infeasible.

5.2 Results

Table 1 provides an overview of the quantitative measures for our three benchmark tasks. According to the data, "collect dirty" has the highest success rate, while "collect wood planks" has the lowest success rate, with none of the trials being successful. The "collect dirt" task is completed within 3600 frames (about 3 minutes), and the hardest task, "collect stone," is completed within 4800 frames (about 4 minutes), which is only a minute longer than the easiest benchmark. Furthermore, the data demonstrate that our method allows the agent to maximize its resources by collecting more items than initially requested.

The discrepancy in success rates is surprising because the agent was able to complete "collect stone" at least once, but it failed to complete "collect wood planks" all three times. We analyzed the recorded agent action and prompting history from ChatGPT qualitatively. Upon examination, we found that the agent struggled with low-level control while in the dense environment that had a lot of trees, as

Task	Ratio of Success	Avg. Time (Frames)	Resources
Collect dirt	2/3	3600	41 dirts, 1 wheat seed, 1 stick
Collect wood planks	0/3	N/A	N/A
Collect stone	1/3	4800	25 dirts, 8 sticks, 1 pickaxe

Table 1: Quantitative evaluation for each benchmark tasks over three trials. The ratio of success shows how many trials (out of 3) are successful. The average time is measured with frames, and we tested with a fixed 30 frames-per-second rate. Resources are observed at the end of the episode (not at the end of our objective; we intentionally run the experiment longer to see what our agent will do).

shown in Appendix A Figure 4. Despite collecting one block of wood, the agent dropped it a couple of frames later, indicating hallucinations in action from STEVE-1. Similar to LLM's behavior, where it confidently contradicts earlier statements, STEVE-1's hallucination can do something different from the instruction (e.g. interrupting wood collection to explore) or actively prevent the action from being completed (e.g. dropping key items). Another hallucinating action STEVE-1 often performs is that the agent can't control the cursor with enough precision to place the crafted item into its inventory, which hinders their ability to complete crafting tasks.

In addition to dropping key action items, STEVE-1 could get stuck in situations where the visual help prompt is not sufficient. In such cases, the agent can encounter a problem as shown in Appendix A Figure 5, where the agent ends up gazing at the sky, and the observation frame doesn't provide any useful information. Even ChatGPT fails to give any helpful solution to the agent to escape from this situation. Consequently, the agent enters an infinite loop of repeatedly asking for help, only to receive a response instructing it to continue with the existing trajectory.

Nevertheless, it has been shown that ChatGPT is capable of generating the best possible action path. During a "collect dirt" episode, ChatGPT came to the conclusion that dirt could be gathered by hand, and therefore skipped the process of crafting a tool, instead opting to collect a dirt block quickly by hand (Figure 7). This suggests that we can implement a verification method to ensure that the trajectories created by LLMs are optimal, which further improves STEVE-1's performance over long-horizon tasks.

6 Limitations

6.1 Hallucination of STEVE-1

During our qualitative evaluation of the recorded episodes, we found that STEVE-1 failed multiple times due to misinputs at low-level controls. In one particular episode, the agent was instructed to "collect stones", but they accidentally dropped the crafted pickaxe without any apparent reason. Though the agent utilized the Help prompt to resume crafting a new pickaxe, they wasted a significant amount of time, and ultimately, the episode timed out. We believe that the cause of this failure lies in the hindsight-relabelled dataset and the behavior cloning objective. The dataset is curated by selecting random segments of gameplay video, with the last frame as the target for all frames in the segment, and STEVE-1 is fine-tuned to merely mimic the actions without any embedded knowledge of any meaningful objectives. This approach implies that the behavior that STEVE-1 learned may not align with the given instructions at all time steps. This is an area that requires further research, and a brute-force solution of fine-tuning with a more diverse set of gameplay videos might work, suggested by the success of other similarly trained large foundational generative models.

6.2 Ineffective Visual Guidance

When the STEVE-1 agent requests help through the help prompt, ChatGPT occasionally provides an ineffective action, resulting in the failure of the episode. This suggests that while a visual prompt can provide ChatGPT with more detailed information, a text-based response may not be sufficient to guide the agent out of a difficult situation. One solution to this could be to enhance the expressiveness of the text-goal embedding space, such as fine-tuning the MineCLIP model.

6.3 The Suboptimality and Instability of LLM Planning

Task sequences planned by ChatGPT were found sometimes suboptimal and unstable in our experiments. ChatGPT showed poor ability in estimating the task time costs and adjusting plans according to the costs (i.e. it instructs to use hand to chop trees rather than craft an axe when prompted to "collect 64 logs"), which is possibly due to the lack of prior knowledge in Minecraft gameplay. A small number of unworkable planning was also found in the task trials due to the randomness of ChatGPT, including skipping intermediate steps and invalid answer formats, which can be corrected by the future *Help* Prompt but took up plenty of time of the trials, influencing the final success.

7 Conclusion

In this work, we introduced an automated chain prompting framework leveraging Large Language Models (LLMs), specifically ChatGPT, to guide a pre-trained autonomous agent, STEVE-1, towards accomplishing long-horizon objectives. Through a review of prior research on planning for autonomous agents, we drew insights from the utilization of LLMs in planning methodologies. Our experimental evaluation, conducted across three benchmark tasks, demonstrates the feasibility of our approach in long-horizon task completion.

Moreover, our investigation revealed notable challenges. Specifically, we identified the presence of a hallucination problem in low-level control STEVE-1, along with irregular inefficacies of the help prompt which might related to visual guidance cues. Future research endeavours may explore novel strategies for mitigating hallucination phenomena and refining the integration of visual guidance mechanisms to bolster the overall performance of autonomous agents guided by LLM-driven prompting frameworks.

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. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24639–24654. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/9c7008aff45b5d8f0973b23e1a22ada0-Paper-Conference.pdf`.

[2] 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.

[3] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6065–6073. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/840. URL `https://doi.org/10.24963/ijcai.2019/840`.

[4] 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. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=rc8o_j8I8PX`.

[5] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2107–2116. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/icarte18a.html`.

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

Figure 3: Agent can't focus on the tree block for the full duration required to finish the collecting action.

[7] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

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

A Notable Gameplay Frames

Figure 4: Agent struggles to finish one action due to the dense environment. We believe, in dense environment, there exist a lot more possible actions in the training dataset, so the agent struggles to learn to complete any one of those actions.

Figure 5: Agent becomes confused and stares at the sky for a long period of time. The help prompt doesn't help much in this case, because the agent's observations do not provide any useful information for ChatGPT to recommend remedy actions.

Figure 6: Open environment from a successful trial for task "collect wood". The agent is able to recognize the action item (wood) and act accordingly (collect wood).

Figure 7: Most successful trial in "collect dirt". ChatGPT succesfully reasoned that dirt can be collected by hand, and prompted STEVE-1 with the optimal action immediately.