Enhancing AI Efficiency: Vision Integration and Multi-Agent Collaboration in the Voyager Framework for Minecraft

Xiaomei Song (3030016042)
Alberto Hojel (3037016737)
Francesco Crivelli (3036606782)
Kunal Agrawal (3035687615)

Group Name: Mind-Guer

Track: Applications

Extended Abstract

This project aims to advance the development of embodied AI agents by leveraging video games as a testing ground, specifically building upon the Voyager framework in Minecraft. We propose enhancing the existing architecture through a combination of state-of-the-art LLM integration, reinforcement learning techniques, and optimized prompt engineering. Our approach focuses on improving long-horizon task completion and skill acquisition, using Minecraft's open-ended environment as a low-risk platform for developing foundational AI capabilities. By optimizing agent behavior through direct gameplay feedback and sophisticated skill library management, we aim to demonstrate significantly improved performance over baseline metrics, particularly in exploration and task completion efficiency. This research contributes to the broader goal of developing more capable and adaptable AI agents, with implications for future applications in multi-agent systems and human-AI collaboration. The open-source nature of our project ensures that our improvements will be accessible to the wider research community, advancing the field of embodied AI.

We are expanding upon the Voyager Paper which created Large Language Agents capable of playing the video game Minecraft. We wanted to take the idea of the project further, and we integrated multi-agent capabilities and added vision agents.

A central focus of the original research was the enhancement of agent behavior through iterative gameplay feedback and the strategic management of a comprehensive skill library. These improvements led to significant advancements in both exploration efficiency and task completion effectiveness compared to baseline models. Our project introduced multi-agent capabilities and incorporated vision-based agents, enriching the interactions and collaborative potential within the Minecraft ecosystem. These enhancements enabled agents to engage in more sophisticated behaviors and adapt to a broader range of scenarios, thereby increasing their overall functionality and resilience.

We used methods similar to those used in the original Voyager paper to evaluate our bot. We will use the distance traveled as one metric, and also use the number of unique items collected by the bot as another metric.

In summary, this project advances the Voyager framework by incorporating AI techniques and expanding its functional scope to include multi-agent interactions and vision-based capabilities. The enhancements in agent performance, evidenced by

improved exploration and task completion metrics, underscore the potential of using video game environments like Minecraft for developing robust and versatile embodied AI agents. As the field continues to evolve, such initiatives are crucial for bridging the gap between simulated and real-world applications, paving the way for more intelligent and interactive AI systems.

1. Introduction

Video games provide the ideal testbed for developing multi-agent AI systems for several compelling reasons:

First, games like Minecraft offer unconstrained virtual environments where AI agents can freely experiment without real-world consequences. The stakes are low - unexpected emergent behaviors during testing can enhance gameplay rather than cause problems. This allows for rapid iteration and testing of fundamental capabilities like autonomy, memory, and social interaction without worrying about reliability issues that would be critical in productivity applications.

Second, games provide built-in data flywheels. AI agents can continuously interact with both human players and other agents, generating rich behavioral data about collaboration, problem-solving, and social dynamics. This creates a natural pipeline for improving the systems through real-world usage rather than synthetic training.

Third, games act as a proving ground for core human-like qualities that will be essential for future applications. The skills needed to be a good gaming companion - emotional intelligence, coherent long-term behavior, collaborative problem-solving, episodic memory - are precisely the foundational capabilities needed for more serious applications like multi-agent workforces or simulated societies.

Finally, games offer a natural path to consumer adoption. Rather than trying to convince people to trust AI agents in high-stakes scenarios, games allow users to organically develop comfort and rapport with AI agents in an entertaining, low-pressure context. This creates a foundation of trust that can later extend to more consequential domains.

In essence, games aren't just a convenient starting point - they're a strategically optimal launchpad for developing the fundamental technologies that will enable more sophisticated multi-agent systems in the future.

Motivation:

The original Voyager paper doesn’t have vision capabilities due to the nature of the GPT-4 API being used at the time. However, in the paper itself, the author says that vision can be integrated in the future. We have decided to actually implement this suggestion and we have added a vision functionality into the agents.

We also have made the appropriate modifications to the codebase to enable multiple agents operating in a world, whereas in the original Voyager paper, there is only a single agent. This is an important addition as it shows the benefits of having another agent and the ability to collaborate and

3. Methodology:

3.1 Multi bots communication:

The multi-bot system significantly expands over the original voyager architecture by enabling multiple bot instances to operate concurrently within the same Minecraft world. At the heart of this system is the Multi-Agent Manager, a central coordination component that handles the initialization and lifecycle management of multiple bot instances. The manager creates separate checkpointing directories for each bot and assigns unique identifiers (like "bot1-alby" and "bot2-france") along with dedicated ports for communication.

System Design:

The system employs an independent process architecture where each bot operates in its own Node.js process through the Mineflayer framework. Communication between bots is facilitated through a REST API interface, with each bot assigned its own port (starting from 3000). To maintain accountability and enable debugging, separate logging directories track each bot's activities, and individual first-person viewports are available for monitoring each bot's perspective in real-time. A crucial aspect of the design is the shared environment state. While bots operate independently, they all exist within the same Minecraft world instance. This means they can observe and interact with blocks and items placed by other bots, creating opportunities for collaborative behavior. The environment maintains a consistent state across all bot instances, ensuring coherent interactions between bots.

Multi-bots Workflow:

In our implementation, we focused on analyzing how agents in Minecraft naturally organize themselves and how quickly this self-organization emerges. Our multi-agent system allows us to examine how agents learn to avoid conflicts, share resources, and develop complementary roles, while also studying whether agents can accelerate their learning by observing each other's behaviors.

Skill sharing represents a key collaborative aspect of the system. While bots maintain their individual skill libraries, they can contribute successful behaviors to a shared skill pool. When one bot successfully completes a task, the associated skill becomes available to other bots, fostering collaborative learning across the bot population. This knowledge-sharing mechanism accelerates the overall learning process and promotes the development of more sophisticated behaviors.

The system incorporates robust environmental awareness mechanisms. Bots actively track each other's positions and activities through environmental observations, while the critic agent evaluates task success by considering the actions of all bots in the environment. This awareness enables bots to coordinate around shared resources and avoid interfering with each other's tasks. Additionally, the system includes sophisticated error recovery capabilities - if one bot encounters issues, the manager can restart it independently while other bots continue their operations, ensuring the system maintains overall progress even in the face of individual bot failures.

This comprehensive multi-bot architecture enhances the capabilities of the original Voyager system by enabling complex collaborative behaviors while maintaining robustness and learning capabilities. Combining independent operation and coordinated interaction creates a flexible and scalable platform for multi-agent learning and task completion.

3.2 Vision agent integration

As outlined in Voyager's original implementation, a significant limitation exists in the realm of multimodal feedback. Specifically, Voyager does not currently support visual perception due to the text-only nature of the available GPT-4 API at its inception. This constraint restricts the framework's ability to perform tasks that require nuanced spatial understanding and visual analysis.

To address this limitation, in this project, we developed and integrated a Vision Agent to enhance spatial reasoning and task efficiency within a multi-agent Minecraft environment. The VisionAgent was designed to analyze images using GPT-4 Vision, extracting critical spatial relationships such as block positions and accessibility and storing these insights in a structured vision_memory.json file. This capability allows the agent to leverage past experiences and adapt strategies for both repetitive and complex tasks through a multi-modal memory system.

The integration of the Vision Agent into the multi-agent workflow was achieved through a modular architecture that allows for seamless communication and data sharing among agents. The Vision Agent is responsible for analyzing visual data captured from the environment and providing actionable insights to other agents in the system. This integration was facilitated by defining clear interfaces for data exchange, enabling the Vision Agent to send its analysis results to other agents, such as the Action Agent, which then uses this information to make informed decisions about task execution.

To facilitate vision-informed task planning, we integrated the Vision Agent into the CurriculumAgent framework. This integration enabled the CurriculumAgent to utilize vision data, enhancing observations and incorporating them into task proposals such as "Mine the nearest block at specific coordinates." By dynamically capturing images as bots navigate the environment, the Vision Agent ensures that visual data is continuously updated and relevant for real-time analysis and logging. Screenshots are systematically saved in a local directory with unique timestamps, and a frame management strategy maintains only the most recent 30 frames to optimize storage and processing efficiency.

The Vision Agent’s insights are shared with other agents, including the Action Agent and Critic Agent, via shared memory files. This shared vision data informs task proposals based on factors like block proximity, accessibility, and clustering, thereby improving the relevance and effectiveness of proposed tasks. The optimized workflow captures and analyzes vision data during bot movement, leveraging these insights for task proposals, execution, and subsequent evaluation. This seamless integration enhances the system's ability to make informed decisions and execute tasks efficiently.

Our modular design ensures that Vision Agent is reusable across different agents and easily updatable, promoting scalability and adaptability within the multi-agent system. This design philosophy supports the implementation of action-perception loops, where agents interact with their environment and refine their spatial understanding dynamically based on vision data.

Multi-Agent Workflow:

In the multi-agent workflow, each agent has a distinct role that contributes to the overall functionality of the system. The primary agents include:

Vision Agent: This agent captures and analyzes visual data from the environment. It identifies objects, assesses their properties, and provides insights to other agents. The Vision Agent acts as the sensory component of the system, enabling other agents to understand their surroundings.

- Input: Receives images from the environment.
- Process: Analyze the image to identify objects, their types, positions, and accessibility.
- Output: Sends structured data (JSON format) to Action Agent, Critic Agent, and curriculum agent

Action Agent: This agent receives insights from the Vision Agent and determines the appropriate actions to take based on the current context. It utilizes information about optimal blocks and other relevant objects to execute tasks such as mining, crafting, or navigating.

- Input: Receives insights from the Vision Agent.
- Process: Determines the appropriate actions to take based on the insights.
- Output: Executes actions in the environment (e.g., mining, crafting) and sends the action results to the Critic Agent for evaluation.

Critic Agent: The Critic Agent evaluates the actions taken by the Action Agent based on the insights provided by the Vision Agent. It assesses the effectiveness of the actions and provides feedback, which can be used to refine the decision-making process of the Action Agent. This feedback loop helps improve the overall performance of the system by ensuring that actions align with the goals of the task.

- Input: Receives action results from the Action Agent.
- Process: Evaluate the effectiveness of the actions taken based on the insights from the Vision Agent.
- Output: Provides feedback to the Action Agent to refine its decision-making process.

Curriculum Agent: Acts as the communication hub and task manager. Integrate Vision Insights into Observations. In the render_observation method, include the Vision Agent's analyzed spatial data. This ensures tasks proposed by the Curriculum Agent are informed by visual information. By integrating Vision Agent, CurriculumAgent can “see” the current environment and tailor tasks that make sense in the immediate context. For example, if the visual input detects trees, CurriculumAgent might prioritize wood-gathering tasks, avoiding irrelevant tasks like mining if no ores are visible.

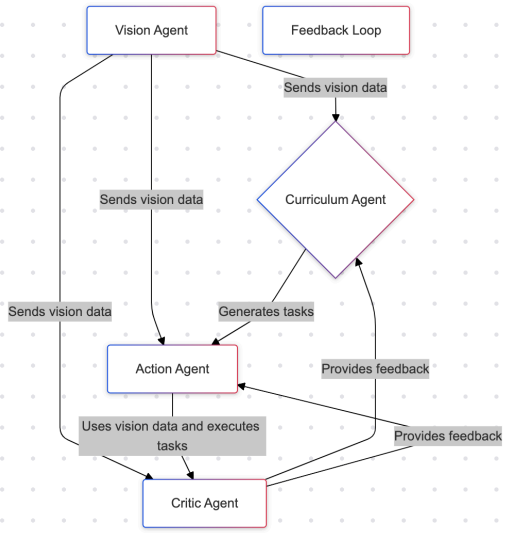
- Input: Receives insights from the Vision Agent and the Action Agent's task execution results.
- Process: Proposes tasks based on the current context, inventory status, and insights from the Vision Agent. It also evaluates the effectiveness of actions taken by the Action Agent through the Critic Agent.
- Output: Sends proposed tasks and context back to the Action Agent.

This lets CurriculumAgent dynamically adapt the task list based on what's actually available or visible in the environment, leading to more efficient exploration and skill-building. Integrate vision agent's insights into task planning. Use the Vision Agent to provide spatial information and long-term memory to provide long-term planning for better task planning.

Agent Interactions and Protocols

Interactions between agents are governed by a message-passing protocol, wherein the Vision Agent sends its analysis results to the Action Agent. The Action Agent then determines the next steps based on these results and executes the corresponding actions. The Critic Agent subsequently evaluates these actions and provides feedback, which the Action Agent uses to refine its decision-making process. This structured interaction ensures that all agents are aligned with the system's goals, promoting coherent and efficient task execution.

In summary, the integration of the Vision Agent into the multi-agent workflow through a modular and communicative architecture has significantly enhanced the system's ability to process visual data, make informed decisions, and continuously improve through feedback. This structured approach facilitates effective collaboration among agents, resulting in a more robust and adaptable AI system.



3.3 Tools and Frameworks:

The development and implementation of the multi-agent system utilized a variety of technologies and frameworks, including:

LangChain: Utilized to manage interactions between agents and facilitate the processing of natural language inputs and outputs.

OpenAI API: Integrated for leveraging large language models (LLMs) to enhance the reasoning capabilities of the agents.

Custom Logging Tools: Developed to capture detailed logs of agent interactions and system performance, enabling thorough analysis and debugging.

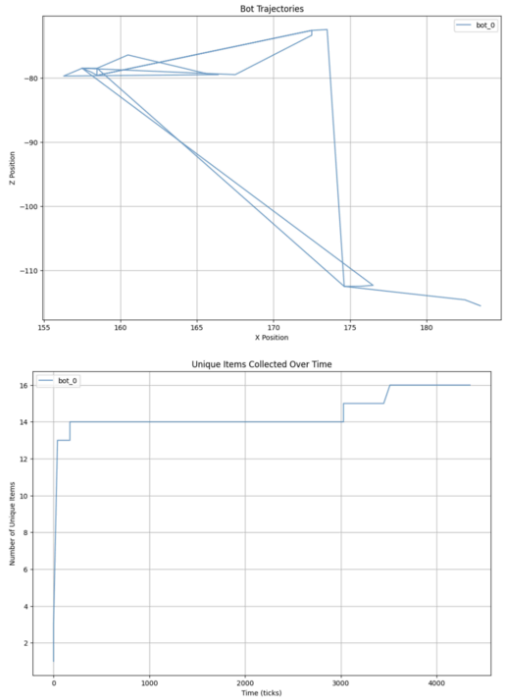
JSON: Used for data interchange between agents, ensuring a structured and easily parsable format for communication.

React, Flask: Front end and back end.

Pandas, Matplotlib, and Seaborn: Popular Python visualization libraries, which we specifically use to generate graphs.

By leveraging these tools and frameworks, the project was able to create a robust and efficient multi-agent system capable of performing complex tasks in a dynamic environment.

4. Experiments and Evaluation



Above are some preliminary results we got from a single bot. We weren't able to have a bot running long enough to get results that we could compare to the Voyager results to get an adequate comparison. To be fair, running the model for long periods of time–with the number of API calls used–did get quite expensive. This is one area (which we elaborate in future work) that can be optimized.

5. Conclusion

This project successfully extended the Voyager framework by integrating vision capabilities and enabling multi-agent workflows within the Minecraft environment. By enhancing the agents with vision processing and facilitating collaboration between multiple agents, We weren't able to have a job running long enough to facilitate a fair comparison with the original Voyager model, and it was also quite expensive to have a job running for a long period of time (It was between $5-$10 for

even short runs of about fifteen minutes). . However, the initial results do seem promising, and expect this model to better than the original model.

The integration of vision agents into multi-agent workflows has substantial implications for the development of embodied AI. Enhanced performance in simulated environments like Minecraft serves as a stepping stone toward more complex real-world applications, including multi-agent workforces, autonomous systems, and human-AI collaborative platforms.

Future Work:

Scalability Enhancements: Expanding the system to support a larger number of agents and more complex tasks, assessing performance and coordination efficiency at scale.

Collaborative Spatial Reasoning: Enable multiple agents to work together on spatial tasks, sharing information about their environments and coordinating actions.

Human-Agent Collaboration: Exploring interactions between AI agents and human players, fostering more natural and intuitive collaboration.

Real-World Applications: Transitioning the multi-agent vision framework from simulated environments to real-world applications, such as robotics and autonomous systems.

Adaptive Learning Mechanisms: Implementing adaptive learning algorithms that allow agents to learn and evolve their strategies based on dynamic environmental changes and collaborative experiences.

Improved Prompting Mechanisms: We didn't get time to fully implement, however, we could explore different prompting strategies such as incorporating DSPy and implementing Chain of Thought, and reducing the number of API calls which reduces monetary constraints.

By pursuing these avenues, future work can build upon the foundational advancements achieved in this project, driving the development of more sophisticated and versatile AI agents. The implications of this research are very exciting, and being able to create agents that can navigate a world and perform tasks can hopefully translate over to the real world, helping the development of robots that are able to navigate their surroundings and perform a variety of tasks.

References

Zhang, K., Yang, Z., & Basar, T. (2020). *Deep Reinforcement Learning for Multi-Agent Systems: A Survey*. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4-24.

Foerster, J., Nardelli, N., Farquhar, G., Torr, P. H., Kohli, P., & Whiteson, S. (2018). *Counterfactual Multi-Agent Policy Gradients*. Advances in Neural Information Processing Systems, 31, 4297-4307.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. AI2-thor: An interactive 3d environment for visual AI. arXiv preprint arXiv:1712.05474, 2017.

Savva, M., Malik, J., Parikh, D., Batra, D., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., and Koltun, V. Habitat: A platform for embodied AI research. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019.

Zhu, Y., Wong, J., Mandlekar, A., and Martín-Martín, R. robosuite: A modular simulation framework and benchmark for robot learning. arXiv preprint arXiv:2009.12293, 2020.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. Advances in Neural Information Processing Systems, 2020.

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (VPT): Learning to act by watching unlabeled online videos. arXiv preprint arXiv:2206.11795, 2022.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. arXiv preprint arXiv:2209.07753, 2022.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. arXiv preprint arXiv:2209.11302, 2022.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D-A., Zhu, Y., and Anandkumar, A. MineDojo: Building open-ended embodied agents with internet-scale knowledge. arXiv preprint arXiv:2206.08853, 2022.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Ichter, B., et al. Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608, 2022.

Significant-gravitas/auto-gpt: An experimental open-source attempt to make GPT-4 fully autonomous, 2023.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629, 2022.

Shinn, N., Labash, B., and Gopinath, A. Reflexion: an autonomous agent with dynamic memory and self-reflection. arXiv preprint arXiv:2303.11366, 2023.

Nottingham, K., Ammanabrolu, P., Suhr, A., Choi, Y., Hajishirzi, H., Singh, S., and Fox, R. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modeling. ARXIV.ORG, 2023.

Wang, Z., Cai, S., Liu, A., Ma, X., and 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.

Driess, D., Xia, F., Sajjadi, M.S.M., 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.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. arXiv preprint arXiv:2301.04104, 2023.

Yuan, H., Zhang, C., Wang, H., Xie, F., Cai, P., Dong, H., and Lu, Z. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. arXiv preprint arXiv:2303.16563, 2023.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Hassabis, D. (2018). *A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play*. Science, 362(6419), 1140-1144.

Peng, X. B., Liu, Y., Wu, Y., Tan, M., Tang, S., Gao, J., ... & Xu, C. (2017). *Multi-Agent Cooperation and Communication with Deep Reinforcement Learning*. arXiv preprint arXiv:1706.02275.

Wei, J., Bosma, M., Zhao, V., Gu, N., Yu, A. W., Lester, B., ... & Le, Q. V. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. Advances in Neural Information Processing Systems, 35, 7701-7713.