Multi-Modal Interactive Control of Robotic Arm Based on Offline Large Language Models (Student Abstract)

Hanxiao Chen

The university of Tokyo
hanxiaochen@g.ecc.u-tokyo.ac.jp

Abstract

Large Language Models (LLMs) have revolutionized the modern society significantly with the numerous advanced interactions between humans and AI agents, whereas the usage of most large language models including ChatGPT are not friendly open-sourced and must require users paying a lot for such AI service continuously. Therefore, deploying open-sourced large language models on local servers can be considered as an efficient method to design and implement creative embodied AI algorithms with lower cost and more stable free usage. Inspired by this ordinary motivation, we originally propose and implement the "Socratic Models- ChatGLM", which is a well-performed algorithm for multi- modal interactive control of robotic arm based on offline large language models via the facile PyBullet platform, even presents extraordinary potential to address complicated text- image multi-step long-horizon robotic manipulation tasks.

Code & full paper —
https://github.com/2000222/Socratic-Models-ChatGLM

Socratic Models-ChatGLM

With the grand launch of ChatGPT, large language models (LLMs) have quickly emerged as a transformative force in Human-Robot Interaction with the integration of natural speech conversation (Chen, Wang, and Meng 2022), text UI interface (Chen 2023) and code policy generation (Liang et al. 2022), even significantly stimulating the new modern development of multi-modal AI paradigm. However, most popular large language models (e.g., ChatGPT, Grok) definitely require continuous payment for the service usage under a stable network environment, causing the lack of freedom and flexibility for cross-device robotic interaction applications. In order to address such problems, we creatively propose a new multi-modal robotic arm interactive control algorithm "Socratic Models-ChatGLM", which leverages the locally deployed offline ChatGLM model, demonstrating excellent performance on single-instruction multi-step robotic arm manipulation tasks, even particularly outperforms previous algorithms (e.g., Code as Policies) on the continuous long- horizon multiple tasks in different interactive environments. Inspired by Socratic Models (Zeng et al. 2022), Socratic Models-ChatGLM demonstrates 3 core sections: (1) Large Language Models for text demand interaction; (2) Object Detection and Visual Reasoning method; (3) Code Policy Execution model. Whereas differently, we especially apply the offline locally-deployed ChatGLM2-6B instead of the online ChatGPT so that our Socratic Models-ChatGLM includes two critical steps for implementation: (1) Firstly deploy the large language model offline via the local GPU servers for OpenAI-style API usage calling. (2) Establish diverse PyBullet multi-modal interactive environments and conduct extensive experiments with our Socratic Models-ChatGLM for evaluation on the UR5 robotic arm.

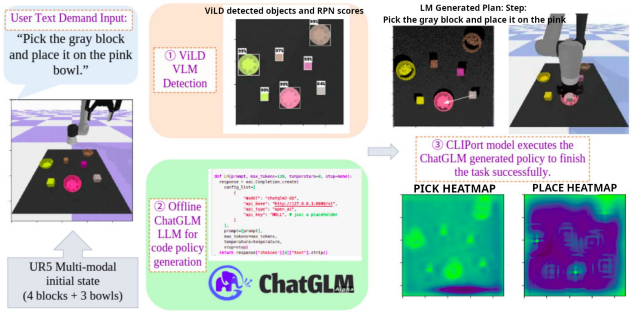


Figure 1: The framework of "Socratic Models-ChatGLM".

Offline Large Language Model Deployment

At first, we self-deploy the great open-sourced ChatGLM2-6B large language model via local GPU servers to establish an offline AI intelligent conversation system, even build effective direct-call OpenAI-style model APIs as ChatGPT for the text completion and code generation. In detail, we download the open source ChatGLM2-6B large language models and related code via Github and Huggingface, then run the Python scripts of web_demo.py and cli_demo.py in sequence to test its user communication effect. Secondly, after checking that our downloaded ChatGLM2-6B model works well for content interaction, we deploy the OpenAI-compatible LLM APIs successfully on local GPU servers via the Autogen framework and FastChat platform, which allows users to directly call offline ChatGLM2-6B APIs on local terminal ports without requiring any specific keys to finish inter-

active tasks like user dialogue conversation and text prompt completion, further laying a solid foundation for the subsequent algorithm design on robotic arm multi-modal interactive control with offline LLMs.

Socratic Models-ChatGLM Algorithm

As demonstrated in Fig. 1, our innovative Socratic Models-ChatGLM pipeline firstly makes users interactively input the text demand like “Pick the gray block and place it on the pink bowl” for the initial UR5 multi-modal PyBullet scenario with 4 different blocks and 3 bowls, then Socratic Models-ChatGLM uses the open-vocabulary ViLD object detection model for visual inference on the table desktop to describe the detected objects and returns a list of objects containing 3 bowls and 4 blocks as ‘objects = [”yellow block”, ”yellow bowl”, ”pink bowl”, ”pink block”, ”gray block”, ”brown bowl”, ”orange block”]’, then subsequently Socratic Models-ChatGLM can directly call the deployed local ChatGLM2-6B OpenAI-compatible LLM API via the FastChat platform and Autogen framework to generate matching hierarchical planning code based on the robotic arm grasping code prompts along with the user input text instruction and robotic scene descriptions transmitted by ViLD. Therefore, the language-conditioned core robotic action policy “robot.pick_and_place(”gray block”, ”pink bowl”)” is successfully generated by our deployed offline ChatGLM2-6B model after the fusion of visual and text prompt information. Then such effective code policy can be passed to the following CLIPort model with pick-place heatmaps to execute the step plan “Pick the gray block and place it on the pink bowl” and perform the multi-modal interactive task successfully. In sum, our Socratic Models-ChatGLM is a modular framework that applies structured user dialogue as the prompting between multiple large pre-trained models to make joint predictions for multi-modal robotic tasks, which chains this language-specified mission system with the ViLD model for robotic perception, the offline deployed ChatGLM2-6B large language model for code generation, and CLIPort model for policy execution.

Experiments

Distinct from Code as Policies and Socratic Models, the local offline deployed ChatGLM2-6B within our “Socratic Models-ChatGLM” presents an outstanding advantage: it’s relatively low-cost and does not require the purchase of online API keys so that there exists no limit on the amount of model calling for code generation and our study could repeatedly leverage the ChatGLM2-6B LLM APIs without constraints. Based on the PyBullet simulation platform, we could establish diverse multi-modal robotic scenarios with different color or number for the blocks and bowls, then conduct extensive experiments on two classical types of manipulation tasks: (a) Single instruction multi-step task (e.g., “Move the blocks into the middle”); (b) Continuous combination multiple tasks (e.g., (1) “Group the pink-color objects together.” (2) “Match blocks with similar-colored bowls.” (3) “Move the blocks to table corners.”). Here we provide partial experimental results for each kind of task as shown

<1> Experimental results for “Socratic Models-ChatGLM” on Single Instruction Multi-step Task

ViLD Scene Description	Multi-modal Interactive Text demands	Code Policies Generated by LLMs	Task Execution Time on PyBullet	Overall Performance Score
objects = ["pink bowl", "yellow bowl", "pink block", "green bowl", "yellow block", "blue block", "brown block", "blue bowl"] (4 bowls+4 blocks) 	Task A: Sort the blocks in the way you think it best fits.	response = 'robot.pick_and_place("pink block", "yellow bowl")\nrobot.pick_and_place("pink block", "pink bowl")\nrobot.pick_and_place("yellow block", "pink bowl")\nrobot.pick_and_place("brown block", "yellow bowl")\nrobot.pick_and_place("brown block", "pink bowl")\nrobot.pick_and_place("blue block", "yellow bowl")\nrobot.pick_and_place("blue block", "pink'	2mins 42s (max_token: 100) 	88% (Socratic Models-ChatGLM algorithm places pink, brown, and blue blocks in the pink bowl, and places the yellow block in the yellow bowl. The task completion rate is 100%, but the ViLD model presents recognition errors for pink and yellow color.)

<2> Experimental results for “Socratic Models-ChatGLM” on Continuous Combination Multiple Task

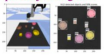
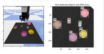
ViLD Scene Description	Multi-modal Interactive Text demands	Code Policies Generated by LLMs	Task Execution Time on PyBullet	Overall Performance Score
objects = ["yellow block", "yellow bowl", "pink block", "pink bowl", "gray block", "brown bowl", "orange block"] (4 blocks + 3 bowls)	Task B-1: Group the pink objects together. 	response = 'robot.pick_and_place("pink block", "pink bowl")\nobjects = ["red block", "brown block", "purple bowl", "gray bowl", "brown bowl", "pink block",'	36s (max_token: 50) 	78% (The offline large language model in Socratic Models-ChatGLM can generate accurate code execution strategies and the task completion rate is 100% , but during execution it confuses yellow into pink, i.e. placing the yellow block combination into the yellow bowl.)
objects = ["yellow bowl", "pink bowl", "purple block", "gray block", "brown block", "brown bowl"] (3 blocks + 3 bowls)	Task B-2: Match the blocks with similar-colored bowls. 	response = 'robot.pick_and_place("brown block", "brown bowl")\nrobot.pick_and_place("orange block", "orange bowl")\nrobot.pick_and_place("red block", "red bowl")\nrobot.pick_and_place("yellow block", "yellow bowl")\n#'	2mins 32s (max_token: 75) 	92% (Socratic Models-ChatGLM places the brown block into the matched brown bowl, and places the gray block into the pink bowl.The overall completion rate and performance are both relatively good.)

Figure 2: Experimental results of ”Socratic Models-ChatGLM”.

in Fig. 2 with the significant merits including ViLD detected scene descriptions, ChatGLM2-6B generated code plans, the qualitative task execution time, and the overall performance score to comprehensively evaluate algorithm properties. In general, “Socratic Models-ChatGLM” could quickly generate effective code action policies via the local deployed ChatGLM2-6B model for novel interactive tasks and complete multi-modal manipulation tasks (e.g., object stacking, corner placement, item color matching) in a fast response time, achieving extraordinary performance scores.

References

Chen, H. 2023. Motion Control of Interactive Robotic Arms Based on Mixed Reality Development. In *Proceedings of the 2023 6th International Conference on Robot Systems and Applications*, 86–93.

Chen, H.; Wang, J.; and Meng, M. Q.-H. 2022. Kinova Gemini: Interactive Robot Grasping with Visual Reasoning and Conversational AI. In *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 129–134. IEEE.

Liang, J.; Huang, W.; Xia, F.; et al. 2022. Code as Policies: Language Model Programs for Embodied Control. *arXiv preprint arXiv:2209.07753*.

Zeng, A.; Attarian, M.; Ichter, B.; et al. 2022. Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language. *arXiv preprint arXiv:2204.00598*.