Cloud Robotics and SE4ML/AI

Shutong Jin

2023-06-30

Research and background

Our interdisciplinary project focuses on achieving a breakthrough in the field of robotic manipulation. By combining expertise in machine learning, robotics, cloud computing, and real-time control, we aim to develop foundational algorithms and system design requirements that can scale networked, distributed, robotic manipulation systems to a large network of cloud-connected robots.

Our vision is to create a network that can continuously collect vast amounts of manipulation training data and dynamically learn from past experiences using federated machine learning with an open platform CloudGrippers that we are currently building.

My part in this project emphasizes the development of methodologies for reasoning about asymptotic scaling behavior, robustness, and transfer-learning of robotic manipulation policies trained at large scale. My machine learning-based approach to robotic manipulation utilizes methods such as deep reinforcement learning and transformer-based learning to enable robots to acquire advanced manipulation skills. As robotic manipulation is currently limited in its ability to scale, I believe our research's success depends on the availability of more data.

My recent work tackling the above issue is "sim to real", referring to transferring a robot control policy or algorithm trained in a simulated environment to a real-world robotic system. The goal of sim-to-real is to reduce the cost and time required for robotic system development and enable faster deployment of robotic systems in the real world. Simulations can be used to generate large amounts of training data and to test and evaluate robotic manipulation algorithms in a safe and controlled environment. My goal is to develop video transformer tools facilitating robotic manipulation with the self-built one-to-one simulator and the real robotic arm platform.

Techniques from Robert's lectures

Blackbox testing in Sim2Real transfer

Blackbox testing is a method of testing software where the internal structure, design, implementation is unknown to the tester. The test cases for blackbox testing are built solely on the software requirements and specifications, aiming for testing functionality and behavior, rather than the code structure. Sim2Real aims at transferring knowledge from simulated to real-world environments and parallelling the concept of blackbox testing in AI engineering. Blackbox testing also aims at examining functionality without knowing the internal structures, similar to the sim-to-real approach tests with the robustness of manipulation policies. In both, the focus is on validating external system behavior while maintaining the system's integrity.

For example, the sim-to-real approach is like training an AI model in a simulated environment and then deploying it on a real-world robotic system. The model's behavior and performance in the real world would be your test. It's just like a blackbox test because we're only concerned with if the robot performs its task, not how the model learned it. While in AI Engineering, the blackbox testing could involve deploying an AI model in a live system and checking if it's performing its tasks correctly, without focusing on how the model was trained or its internal workings. In both cases, the output is evaluated without delving into the learning process or internal functioning of the model.

Cloud Robotics and SE4ML/AI

Cloud Robotics' foundational concepts and methodologies are closely intertwined with Software Engineering for Machine Learning and Artificial Intelligence (SE4ML/AI), a growing field that places an emphasis on effective techniques for the design, implementation, and maintenance of software systems that utilize machine learning and AI technologies. SE4ML looks to address unique challenges that arise from the complexity of developing and deploying machine learning models in production environments, which is directly in line with your research on the development of scalable, robust and transferable robotic manipulation policies.

As an open platform for networked, distributed, cloud-connected robots, CloudGrippers needs robust software engineering principles to manage the complexity of the system, ensuring high scalability, reliability, and maintainability. These principles are crucial to handle the continuous collection, processing, and learning from vast amounts of manipulation training data. SE4ML/AI focuses on handling this exact kind of complexity by identifying best practices, standards, and methods to make machine learning models robust, interpretable, and able to run efficiently and reliably in a production environment.

Here are some thoughts on how to merge cloud robotics platform and SE4ML. First, on the system scalability front, SE4ML/AI will play a pivotal role. As the platform grows, so does the complexity of managing multiple machine learning models running simultaneously across a network of cloud-connected robots. Software The dynamic nature of CloudGrippers, which continually collects and learns from vast amounts of data, makes it vital to ensure that the deployed models are reliable and robust. To achieve this, we can leverage various SE4ML/AI methodologies such as Continuous Integration/Continuous Deployment (CI/CD) for machine learning models.

Techniques from guest lectures

Generally, a "devbot" is an abbreviation for "developer bot", which are typically automated software applications that can perform a range of tasks that aid developers in their work. ## Devbots for Robotic Network Management

The notion of devbots as automated entities aligns perfectly with the broad objective of my research. A critical aspect of my project is the automated learning and improvement of robotic manipulation policies through federated machine learning. This concept parallels the principle behind devbots, which is to automate routine tasks to improve overall efficiency. In my research, I could leverage the idea of devbots by developing AI-powered bots that automatically manage and coordinate the data sharing and learning processes across the cloud network. These devbots could be responsible for aggregating training data, distributing machine learning tasks, and even implementing learned policies across the distributed robotic systems. Such automation would streamline the operation of the CloudGrippers platform, improve its scalability, and potentially accelerate the learning process of the robotic network.

Devbots in Robotic Policy Testing

As devbots are often employed in software development for testing and debugging purposes. In the context of my research, I could conceptualize devbots as automated testing agents that rigorously test the performance and robustness of the learned robotic manipulation policies in simulated and real-world scenarios. These devbots could conduct an array of tests to assess how well the learned policies generalize across different robots, environments, and tasks. Moreover, they could help in detecting and debugging any issues in the implementation of the manipulation policies across the robotic network. This form of automation could dramatically enhance the reliability of my platform and ensure a high-quality, consistent performance of the distributed robotic systems.

In summary, by automating processes and testing software solutions, devbots can enhance the scalability, efficiency, and reliability of my CloudGrippers platform, aiding in the realization of my vision for a large-scale, networked robotic system.

Chosen topics

Automated Software Testing

My understanding with automated software testing is this represents an integral part of modern software development life cycles. It involves employing various software tools to execute tests on another software application, with the overarching goal of ensuring the software behaves as expected under varying conditions and scenarios. More so, automated software testing assists in identifying possible bugs, vulnerabilities, or any issues that might degrade the software's performance, disrupt its intended functionality, or expose it to potential security risks. A crucial use case of automated software testing lies in regression testing, where it facilitates the validation of newly implemented changes without disrupting or breaking the existing functionalities.

The advent of cloud computing, machine learning, and robotic process automation has significantly widened the scope of automated software testing. From my perspective as a researcher, automated software testing has considerable potential, particularly when applied in testing robotic manipulation policies. As the network of cloud-based robots scales up and the complexity of tasks assigned to these robots increase, the necessity for efficient, automated testing frameworks becomes increasingly important.

One exciting opportunity in this realm is to develop machine learning-driven testing frameworks that can intelligently stress-test the learned policies across a broad range of scenarios. Such frameworks can leverage machine learning to predict potential failure points, enhance the efficiency of testing processes, and consequently contribute to creating more robust and reliable robotic systems. The paper "Machine Learning Testing: Survey, Landscapes and Horizons" by Zhang et al., [1] presents a comprehensive survey on machine learning testing. This research, which aligns closely with my work on scalable robotic manipulation systems, emphasizes testing properties like correctness, robustness, and fairness, alongside aspects such as test generation and test evaluation. The application scenarios they discuss, particularly those that involve autonomous systems, resonate directly with my work on autonomous, ML-driven robotics.

Human-Computer Interaction (HCI)

Human-Computer Interaction (HCI) is an interdisciplinary field that integrates computer science with behavioral sciences to make computers more usable and receptive to users' needs. It involves the study and planning of the interaction between users (humans) and computers, focusing not only on interfaces but also on other aspects of the user experience.

In the context of my research on cloud-connected robotics, I believe HCI plays an instrumental role in making the CloudGrippers platform more user-friendly and universally accessible. Ensuring the system is intuitive and straightforward to use for a wide range of users, from technicians to researchers, is paramount.

An area of opportunity that emerges in this context is the design and development of intuitive, efficient, and adaptive user interfaces. These interfaces should enable users to seamlessly interact with the cloud-connected robots and to monitor and manage the robotic manipulation tasks. More specifically, they should provide users easy access to real-time data while also offering them the ability to adjust manipulation policies or provide new learning examples when necessary.

The paper "A Survey of Human-Centered Evaluations in Human-Centered Machine Learning" by Sperrle et al., [2] delves into the integration of HCI and machine learning, providing an insightful perspective that could help guide the design of the CloudGrippers platform's user interface. They highlight the need for more human-centered evaluation methods in machine learning, which can provide significant value to my research, particularly when creating interfaces that non-expert users can easily navigate and use.

In conclusion, the integration of both automated software testing and HCI within my PhD project could bring substantial benefits, enhancing the robustness, scalability, and user experience of the CloudGrippers platform. Such advancements will ensure the platform remains resilient in the face of growing complexity, adaptable to user needs, and user-friendly, thus augmenting its potential impact in the field of robotics and beyond.

Conclusion

AI and Machine Learning (ML) are dramatically shaping software engineering, a trend towards ML/AI Engineering I foresee intensifying in future years. ML/AI Engineering involves the integration of software engineering principles in the creation and maintenance of ML/AI systems, transforming conventional software engineering to include managing data pipelines, ML model development, and deployment.

For my robotic manipulation systems research, ML/AI Engineering is critical. Efficient data management, quick ML model prototyping, and seamless integration with broader systems are becoming essential in deploying scalable cloud-connected robotic networks like CloudGrippers. Adhering to ML/AI Engineering principles ensures system robustness, reliability, and scalability.

Looking forward, I predict a surge in demand for ML/AI Engineering expertise in both academia and industry. Continued research will tackle ML/AI system complexities like interpretability and fairness, while industry-wide adoption will necessitate proficient ML/AI engineers. I also anticipate AI/ML democratization with more user-friendly tools, driving innovation and ethical, transparent system creation.

In essence, the future of software engineering is rooted in ML/AI Engineering. As we explore AI/ML's potential, our software engineering practices must evolve to address their unique challenges and opportunities, marking an exciting path for my research and career.

References

[1] Zhang JM, Harman M, Ma L, Liu Y. Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering. 2020 Feb 17;48(1):1-36.

[2] Sperrle F, El-Assady M, Guo G, Borgo R, Chau DH, Endert A, Keim D. A Survey of Human-Centered Evaluations in Human-Centered Machine Learning. InComputer Graphics Forum 2021 Jun (Vol. 40, No. 3, pp. 543-568).