Proceedings of the Canadian Society for Mechanical Engineering International Congress
32nd Annual Conference of the Computational Fluid Dynamics Society of Canada
Canadian Society of Rheology Symposium
CSME-CFDSC-CSR 2025
May 25–28, 2025, Montréal, Québec, Canada

Development of AI-Driven Universal Robot with Vision Perception for Robotic Manipulation

Md Shafakat Masud[1,2],Xiaolong Liu[1], Kenneth Matinez[1],Riley Workman[1], Jing Ren[1], Haoxiang Lang[1*]
[1]Department of Mechatronics and Manufacturing Engineering, Ontario Tech University, Oshawa, Canada
[2]Department of Nuclear Science and Engineering, Military Institute of Science and Technology, Dhaka, Bangladesh

Abstract—This paper presents an AI-driven robotic manipulation framework integrating vision-based perception, natural language processing (NLP), and reinforcement learning for autonomous object sorting. The system employs YOLOv7 for real-time object detection, a state-machine-based execution framework, and ChatGPT-4o for task planning. The research follows a dual-phase evaluation, conducting simulations in Gazebo using KUKA iiwa14, ABB IRB120, and UR5, followed by real-world experiments with a UR5 robotic arm. Performance is assessed based on task completion time, success rate, and error analysis, including position and orientation accuracy. Simulation results indicate that KUKA iiwa14 consistently achieved the shortest execution times, averaging 56.34s for Task 1, while UR5 exhibited higher variability, peaking at 139.69s in some sessions. The average success rate across tasks exceeded 85%, though UR5 recorded increased RMSE in orientation (5.78 in Task 1), highlighting challenges in fine manipulation. Failure analysis identified motion planning errors as the dominant cause of failures in Task 1 (50%), while classification errors (75%) and sequencing issues (80%) were prevalent in Tasks 2 and 3, respectively. Experimental validation confirmed the system's feasibility, though real-world trials revealed higher execution times and reduced success rates due to environmental uncertainties. Additionally, reliance on ChatGPT-4o introduces challenges related to internet dependency and API costs. Future work will focus on transitioning to an open-source LLM (LLAMA) for local execution and further optimizing reinforcement learning strategies for real-time adaptability and improved robotic precision.

Keywords-chatgpt-4o; UR5, Robotic Arms; YOLOv7; KUKA iiwa14; ABB IRB120.

I. INTRODUCTION

Robotic arms have become indispensable in modern automation, playing a pivotal role in diverse sectors ranging from industrial manufacturing to precision-driven fields like surgery and hazardous material handling [1]. Universal Robots (UR) have emerged as a leading platform in the field of industrial and collaborative robotics, providing highly adaptable and programmable robotic arms that can be deployed across various industries. At the core of robotic system integration lies the Unified Robot Description Format (URDF), an XML-based modeling framework that defines a robot's physical attributes, kinematic structure, joint constraints, and sensors within simulation and control environments [2]. However, despite its advantages, URDF lacks native AI capabilities, particularly in handling perception-based tasks and autonomous decision-making. This limitation hinders robotic arms from operating effectively in dynamic, real-world environments where adaptability and intelligence are crucial.[3]

Additionally, programming robots remains a complex and time-consuming task that demands expert knowledge in robotic control languages. This limits their accessibility, particularly for applications that necessitate frequent task reconfigurations, such as small-scale manufacturing or service robotics, where adaptability is essential [4]. Another significant challenge in robotic manipulation is the lack of perceptual intelligence. While computer vision algorithms have made strides in object detection and classification, the seamless integration of these capabilities into robotic arms for real-time manipulation remains a complex issue [5].

To address these limitations, this research focuses on developing an AI-driven Universal Robot model that integrates vision perception and natural language processing (NLP) for enhanced robotic manipulation. The proposed system incorporates a Vision Perception Module, utilizing YOLOv7 for high-speed and accurate real-time object detection, allowing the robot to recognize, classify, and interact with objects dynamically within its workspace. Additionally, an NLP-based Command System powered by ChatGPT API is integrated, enabling the robotic arm to understand, interpret, and execute tasks based on natural language commands without requiring manual programming.

II. LITERATURE REVIEW

The integration of Artificial Intelligence (AI) with robotic manipulation has revolutionized the field of automation. Pan et al. proposed a deep reinforcement learning (DRL)-based

control mechanism, comparing Q-learning, Proximal Policy Optimization (PPO), and a hybrid DRL model for robotic manipulation. Their results demonstrated that DRL significantly improved accuracy and reduced execution delay compared to traditional methods. However, their approach lacked real-world sensor integration, limiting its adaptability in unstructured environments [6].

Similarly, Jin et al. introduced RobotGPT in their study, which proposed a self-correcting mechanism for refining AI-generated robotic commands, achieving a 91.5% execution success rate in pick-and-place tasks. However, RobotGPT struggled with iterative refinement, often requiring manual debugging to ensure code correctness [7]. Vemprala et al. further explored ChatGPT's role in robotic interaction, proposing a prompt engineering framework that enabled high-level task execution through structured dialogue-based commands. However, ChatGPT lacked embedded physics reasoning [8]. To tackle this, Gargioni and Fogli introduced Blockly-ChatGPT, a visual programming interface that allows non-experts to program robotic tasks using natural language. Their system successfully bridged the gap between text-based AI reasoning and executable robotic code, but relied on predefined command sets, limiting flexibility in novel tasks [9].

Sekkat et al. implemented Deep Deterministic Policy Gradient (DDPG) for robotic arm control, integrating YOLOv5 for vision-based grasping. Their system achieved a 95.5% grasp success rate, outperforming traditional inverse kinematics. However, the high computational cost of training remained a major limitation [10]. Moreover, Matulis and Harvey explored digital twin frameworks for reinforcement learning, proposing a Unity-based simulation environment to train robotic arms before real-world deployment. Their findings showed that digital twins significantly reduced training costs and hardware wear, but failed to account for real-world sensor noise, leading to performance degradation [11].

Vision-based robotic perception is critical for enhancing environmental awareness, object manipulation, and scene understanding. Guo et al. developed a YOLOv5-based waste classification system, integrating a robotic arm for automated sorting of recyclable materials. Their system achieved 99% classification accuracy. However, their robotic control relied on predefined grasping motions, limiting adaptability in non-uniform object arrangements[12]. Hong et al. explored 3D point cloud-based robotic perception, introducing ResVoteNet for high-precision grasp detection. However, the computational cost of 3D point cloud processing limited its real-time applicability [13].

Therefore, several challenges remain, including high computational costs in reinforcement learning, real-time adaptability in vision-based systems, and URDF standardization issues. Which is why, there is need for research that focus on developing hybrid AI frameworks that combine reinforcement learning, vision-based perception, and NLP-driven task execution.

III. METHODOLOGY

A. *Overview of the AI-Driven Robotic Framework*

The proposed AI-driven robotic framework introduces a multi-layered decision-making system that integrates vision-based perception, natural language processing (NLP), and reinforcement learning to enhance adaptive robotic manipulation, addressing the limitations of traditional rule-based robotic systems. The framework, structured around a state-machine architecture, dynamically manages user commands, sensory perception, motion planning, and error recovery, ensuring real-time adaptability in unstructured environments. The vision-based perception module, powered by YOLOv7, enables the robot to identify, localize, and interact with objects dynamically, allowing for precise grasping and real-time scene interpretation. The NLP module, utilizing ChatGPT-4o, translates human instructions into executable robotic actions, dynamically adjusting tasks based on environmental inputs and contextual changes. This interaction between NLP and perception facilitates error correction, learning-based adaptation, and seamless human-robot collaboration. Furthermore, the gripper control system, integrated with force torque sensors, enables real-time feedback mechanisms, allowing the robot to detect grasping failures, reconfigure its motion plan, and retry executions dynamically before transitioning to real-world execution.

B. *Vision Based Object Detection*

To achieve autonomous robotic manipulation, real-time environmental perception is essential. This module integrates computer vision techniques with a ZED stereo camera to provide

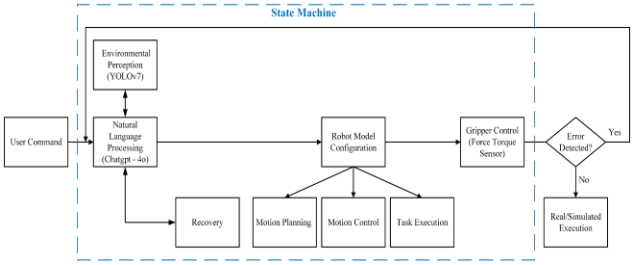

Figure 1. Robot Manipulation Framework

Figure. 2 : Experimental Setup of Robotic Arm Picking a Plastic Bottle

depth information and object localization. The YOLOv7 model is employed for object classification and recognition, while Semi-Global Block Matching (SGBM) and Principal Component Analysis (PCA) are used for depth estimation and localization, respectively.

Once an object is detected, the system computes its depth using the SGBM disparity map and generates a point cloud data. PCA is applied to fit 3D bounding boxes, yielding object positions ($Pcamera$) and orientations ($Rcamera$) in the camera coordinate frame. To align object coordinates with the robotic workspace, a coordinate transformation is applied using a rotation matrix (R), computed as [14]:

$$R = Rcamera \cdot Rinit \quad (1)$$

where $Rcamera$ represents the camera's orientation, and $Rinit$ defines the initial transformation between the camera and the robot's coordinate system. The object's position in the robot's frame ($Probot$) is computed as follows [14]:

$$Protated = R \cdot Pcamera \quad (2)$$

$$Probot = Protated + t \quad (3)$$

where t is the camera's position vector. Similarly, the object's rotation in the robot coordinate system ($Rrobot$) is given by [14]:

$$Rrobot = R \cdot Rcamera \quad (4)$$

The system ultimately outputs object names along with their transformed coordinates, allowing the robotic arm to execute grasping and interaction tasks accurately. This object detection and localization framework serves as a critical foundation for real-time robotic manipulation, enabling the robot to perceive, identify, and engage with its environment in an adaptive manner.

C. Natural Language Processing for AI-based Task Execution

The integration of natural language processing (NLP) with robotic task execution enhances human-robot interaction by enabling intuitive command interpretation and dynamic decision-making. At the core of this framework is a context update mechanism, which acts as a central processing unit for integrating user input, vision-based object detection, and task execution feedback. When a user command is received, the system first updates its internal state with information from the YOLOv7-based perception module, which provides a real-time image feed and object localization data. The updated context is then forwarded to ChatGPT-4o, which interprets the command, refines task parameters, and generates a structured response defining the task function, object label, world coordinates, and execution state. When the robotic system initiates a task, its execution is continuously monitored through a feedback loop that assesses task completion status. If the execution fails or an error is detected, ChatGPT-4o reinterprets the context and refines the task execution strategy, allowing the system to adapt dynamically.

As shown in the Figure. 4, the robotic system transmits critical data to ChatGPT-4o, including real-time images, detected object lists with coordinates, and execution results. In return, it receives a well-defined command structure with task-specific details, world-space object positioning, and task state updates.

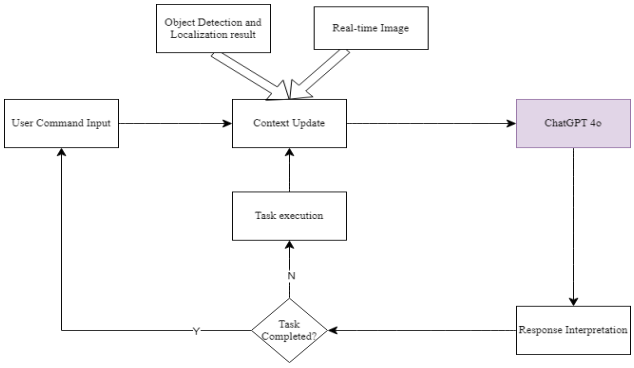

Figure. 3:GPT interaction Flow

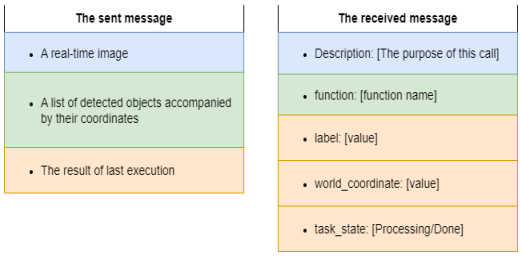

Figure. 4: Message Format

IV. Results and Discussion

To evaluate the AI-driven robotic manipulation framework, a comprehensive simulation study was conducted in Gazebo, involving three different robotic arms—KUKA iiwa14, ABB IRB120, and UR5. The robotic system was tested across three distinct tasks:

- Task 1 (Wooden Ball): Detecting, grasping, and placing a wooden ball in the recycling bin.
- Task 2 (Water Bottle): Identifying, securely grasping, and placing a plastic water bottle in the recycling bin.
- Task 3 (Carrot): Recognizing, gripping, and moving a carrot to the non-recyclable bin.

These tasks were chosen to evaluate the robot's ability to handle objects with varying shapes, materials, and placement precision requirements.

A. Simulation Results

Task execution time serves as a critical metric in pick-and-place operations where speed and precision are equally important. The completion times of KUKA iiwa14, ABB IRB120, and UR5 were recorded across ten independent sessions, with each session representing a full execution cycle of detecting, grasping, and placing an object into the designated bin. The results, as depicted in Figure. 6, illustrate significant variations in execution efficiency among the three robotic platforms.

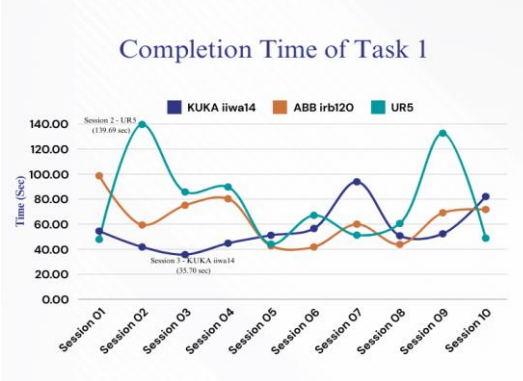

Figure 5: Completion Time of Task 1 for Different Robots Across Sessions

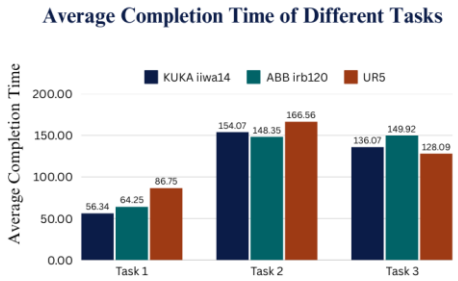

Figure 6: Average Completion Time of Different Tasks

From Figure 5, UR5 exhibited the highest variability, with completion times ranging from 69.41 seconds in Session 7 to 139.69 seconds in Session 2, reflecting the iterative nature of reinforcement learning-based control. In contrast, KUKA iiwa14 maintained consistently lower execution times, with values ranging from 35.70 to 92.83 seconds, benefiting from structured motion planning. ABB IRB120 demonstrated moderate performance, with execution times fluctuating between 60.42 and 100.57 seconds across different trials.

Expanding this analysis across all tasks (Figure. 6), UR5 had the longest execution times, averaging 86.75 seconds for Task 1 (Wooden Ball), 166.56 seconds for Task 2 (Water Bottle), and 128.09 seconds for Task 3 (Carrot). KUKA iiwa14 was the most efficient, completing Task 1 with an average of 56.34 seconds, Task 2 in 154.07 seconds, and Task 3 in 136.07 seconds.

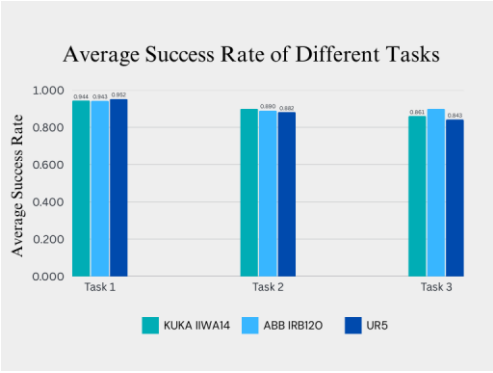

Figure 7: Average Success Rate of Different Tasks

Beyond efficiency, the success rate is a fundamental indicator of a robotic system's reliability in task execution. The success rate across different robotic platforms was analyzed, as shown in Figure 8. KUKA iiwa14 consistently exhibited the highest success rates, achieving 94.4% in Task 1, 89.0% in Task 2, and 86.1% in Task 3. ABB IRB120 and UR5 followed closely behind, with ABB IRB120 averaging 95.2% in Task 1 and 88.2% in Task 2, while UR5 showed a slight decline in Task 3, dropping to 84.3% success.

Interestingly, while ABB IRB120 and UR5 displayed competitive success rates in simpler tasks (Task 1), their performance gradually declined as task complexity increased. Task 3, which involved complex grasping and object manipulation, showed the highest failure rates across all robots.

The average orientation errors recorded during the simulation are presented in Figure. 8, with KUKA iiwa14 demonstrating the lowest deviation across all tasks. For Task 1, KUKA iiwa14 had the lowest average orientation error at 2.72°, followed by ABB IRB120 at 4.81°, and UR5 at 5.96°. The trend persisted across Task 2, where KUKA iiwa14 maintained a low error of 3.77°, whereas ABB IRB120 and UR5 recorded 4.48° and 4.27°, respectively. For Task 3, ABB IRB120 exhibited the highest orientation error at 5.22°, while UR5 slightly improved to 4.68°. Additionally, Root Mean Square Error (RMSE) for orientation accuracy was computed to assess the consistency of robotic motion across different tasks. As shown in Fig. 9, UR5 consistently exhibited the highest RMSE values, indicating greater fluctuations in its orientation stability. The RMSE for UR5 peaked at 5.78° for Task 1 (Wooden Ball), 5.14° for Task 2 (Water Bottle), and 5.36° for Task 3 (Carrot). In contrast, KUKA iiwa14 maintained the lowest RMSE values, demonstrating superior control precision due to its structured motion planning and predefined kinematic constraints.

Failure case analysis provides crucial insights into the factors affecting unsuccessful task executions and helps identify areas for improvement in motion planning, object classification, and NLP-driven task execution. As shown in Figure. 10, motion planning errors accounted for 50% of failures in Task 1 (Wooden Ball), making it the most significant failure cause in this task. GPT execution failures contributed to 30% of the failures, reflecting challenges in translating natural language instructions into precise robotic actions.

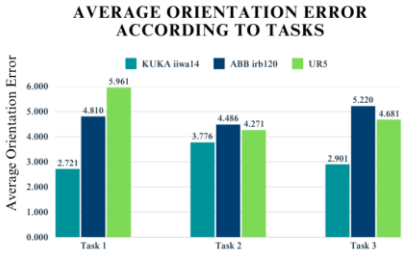

Figure. 8: Average Orientation Error Across Different Tasks

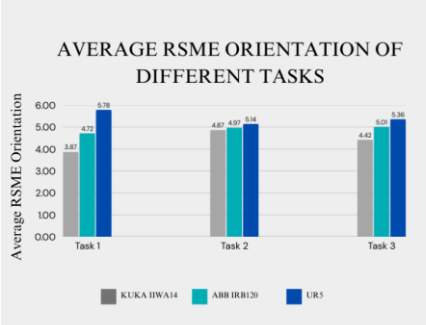

Figure 9: Average RSME Orientation of Different Tasks

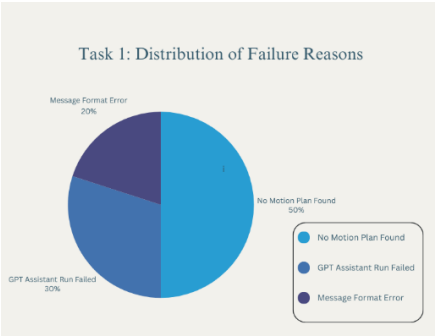

Figure 10: Distribution of Failure Reason for Task 1

In Task 2 (Water Bottle), classification errors emerged as the dominant failure source, accounting for 75% of failures, as illustrated in Figure. 11. This indicates that YOLOv7 and depth-based localization struggled with the irregular positioning of the water bottle, leading to incorrect grasping attempts. For Task 3 (Carrot), incorrect execution order was responsible for 80% of failures, making it the most significant challenge in this task, as presented in Fig. 12. These failures primarily stemmed from errors in task sequencing and NLP-based execution planning, where the system failed to follow the correct sequence of detecting, grasping, and placing the object. GPT misinterpretations contributed to 10% of failures, reinforcing the need for better contextual understanding in NLP-driven robotic task execution.

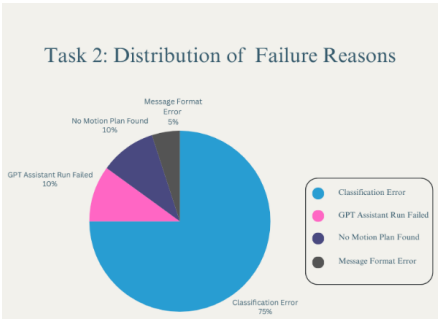

Figure 11: Distribution of Failure Reason for Task 2

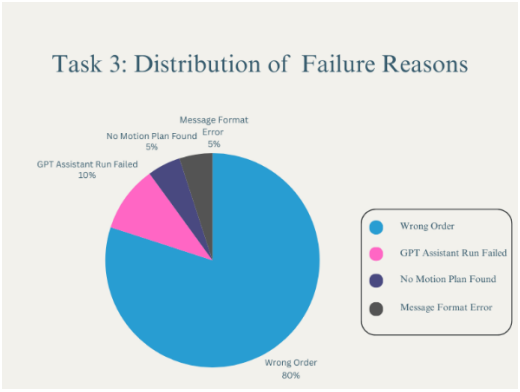

Figure 12: Distribution of Failure Reason for Task 3

B. *Experimental Results*

The experimental validation of the AI-driven robotic framework was conducted using the UR5 robotic arm, performing object categorization and sorting tasks similar to the simulation environment. The experimental results, summarized in Table I and II, reveal notable differences between the real-world and simulated performances.

The highest recorded success rate in an individual session was 90%, while the lowest dropped to 50% in Session 8. The variation in success rates can be attributed to environmental uncertainties, such as lighting conditions affecting vision-based perception and minor calibration offsets in the physical setup. In terms of execution time, the average task completion time was 187.04 seconds, significantly higher than KUKA iiwa14's simulated performance but comparable to UR5's Gazebo execution times. Notably, Session 3 recorded the highest execution time at 230.70 seconds, which aligns with instances where grasping failures necessitated task retries.

Table. I: UR5 Categorization and Sorting task

Session	Success Rate	Average Completion Time
1	0.90	210.40
2	0.80	182.80
3	0.70	230.70
4	0.80	175.78
5	0.80	160.15
6	0.60	150.53
7	0.70	203.83
8	0.50	192.73
9	0.90	181.33
10	0.80	182.12
Avg	**0.75**	**187.04**

Table. III: UR5 Categorization and Sorting task (Continued)

Session	Average Position Error	RMSE Position	Average Orientation Error	RMSE Orientation
1	0.010	0.015	2.20	3.70
2	0.012	0.018	2.50	3.90
3	0.011	0.016	2.30	3.63
4	0.009	0.014	4.40	5.13
5	0.008	0.012	2.10	3.63
6	0.013	0.019	4.60	5.13
7	0.012	0.018	4.50	5.13
8	0.015	0.020	2.70	3.63
9	0.010	0.015	4.20	5.13
10	0.009	0.013	2.10	3.63
Avg	**0.0109**	**0.0160**	**3.16**	**4.264**

C. Discussion of Experimental Results

The experimental results highlight the performance boundaries of AI-driven robotic manipulation under real-world constraints. While the system maintained an average success rate of 75%, this figure marks a modest decline from simulation results, primarily due to uncontrolled factors such as ambient lighting variations and minor misalignments in the camera calibration affecting vision-based localization. Notably, RMSE in orientation (4.264°) and position (0.016 m) reflect the compounded effect of sensor noise and imperfect trajectory execution, particularly during fine alignment phases of the grasping routine. The experiment thus validates the framework's generalizability, while exposing the nuanced gap between simulated precision and physical robustness.

V. Conclusion and Future Works

This study presented an AI-driven robotic manipulation framework that integrates vision perception, NLP to enable a UR5 robotic arm to autonomously detect, categorize, and sort objects. Simulation results demonstrated that KUKA iiwa14 exhibited the fastest execution times due to structured motion planning, whereas UR5, utilizing reinforcement learning, showed higher variability in task completion but maintained adaptability in unstructured conditions. The UR5 achieved an average success rate of 75%, lower than its simulation counterpart, primarily due to environmental factors affecting object detection and grasp stability. Execution times in physical trials were longer than in simulation, reflecting real-world challenges such as motor delays and calibration deviations.

The current system's reliance on ChatGPT poses limitations due to its dependence on an internet connection and the need for API token purchases. To enhance autonomy and reduce operational costs, future work will focus on transitioning to LLAMA, an open-source alternative that allows local deployment without recurring expenses.

Acknowledgment

This research was funded by the Natural Science and Engineering Research Council of Canada (NSERC) through the Discovery Grant program (RGPIN-2024-06516).

References

[1] M. E. Moran, "Evolution of robotic arms," J Robotic Surg, vol. 1, no. 2, pp. 103–111, Jul. 2007, doi: 10.1007/s11701-006-0002-x.

[2] M. Chignoli, J.-J. Slotine, P. M. Wensing, and S. Kim, "URDF+: An Enhanced URDF for Robots with Kinematic Loops," in 2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids), Nancy, France: IEEE, Nov. 2024, pp. 197–204. doi: 10.1109/Humanoids58906.2024.10769903.

[3] M. Ivanou, S. Mikhel, and S. Savin, "Robot description formats and approaches: Review," in 2021 International Conference "Nonlinearity, Information and Robotics" (NIR), Innopolis, Russian Federation: IEEE, Aug. 2021, pp. 1–5. doi: 10.1109/NIR52917.2021.9666120.

[4] P. George, C.-T. Cheng, T. Y. Pang, and K. Neville, "Task Complexity and the Skills Dilemma in the Programming and Control of Collaborative Robots for Manufacturing," Applied Sciences, vol. 13, no. 7, p. 4635, Apr. 2023, doi: 10.3390/app13074635.

[5] Z. Xia, Z. Deng, B. Fang, Y. Yang, and F. Sun, "A review on sensory perception for dexterous robotic manipulation," International Journal of Advanced Robotic Systems, vol. 19, no. 2, p. 17298806221095974, Mar. 2022, doi: 10.1177/17298806221095974.

[6] Z. Pan, J. Zhou, Q. Fan, Z. Feng, X. Gao, and M. Su, "Robotic Control Mechanism Based on Deep Reinforcement Learning," in 2023 2nd International Symposium on Control Engineering and Robotics (ISCER), Hangzhou, China: IEEE, Feb. 2023, pp. 70–74. doi: 10.1109/ISCER58777.2023.00018.

[7] Y. Jin et al., "RobotGPT: Robot Manipulation Learning From ChatGPT," IEEE Robot. Autom. Lett., vol. 9, no. 3, pp. 2543–2550, Mar. 2024, doi: 10.1109/LRA.2024.3357432.

[8] S. H. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "ChatGPT for Robotics: Design Principles and Model Abilities," IEEE Access, vol. 12, pp. 55682–55696, 2024, doi: 10.1109/ACCESS.2024.3387941.

[9] L. Gargioni and D. Fogli, "Integrating ChatGPT with Blockly for End-User Development of Robot Tasks," in Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, Boulder CO USA: ACM, Mar. 2024, pp. 478–482. doi: 10.1145/3610978.3640653.

[10] H. Sekkat, S. Tigani, R. Saadane, and A. Chehri, "Vision-Based Robotic Arm Control Algorithm Using Deep Reinforcement Learning for Autonomous Objects Grasping," Applied Sciences, vol. 11, no. 17, p. 7917, Aug. 2021, doi: 10.3390/app11177917.

[11] M. Matulis and C. Harvey, "A robot arm digital twin utilising reinforcement learning," Computers & Graphics, vol. 95, pp. 106–114, Apr. 2021, doi: 10.1016/j.cag.2021.01.011.

[12] Y. Guo, H. Kameda, and B. Wu, "Design of Household Robotic Arm System to sort Recyclable Resources based on Deep Learning," in 2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Abu Dhabi, United Arab Emirates: IEEE, Nov. 2023, pp. 0537–0542. doi: 10.1109/DASC/PiCom/CBDCom/Cy59711.2023.10361472.

[13] S. Hong et al., "Research of robotic arm control system based on deep learning and 3D point cloud target detection algorithm," in 2022 IEEE 5th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), Shenyang, China: IEEE, Nov. 2022, pp. 217–221. doi: 10.1109/AUTEEE56487.2022.9994321.

[14] L. Cao, X. Zheng, and L. Fang, "The Semantic Segmentation of Standing Tree Images Based on the Yolo V7 Deep Learning Algorithm," Electronics, vol. 12, no. 4, p. 929, Feb. 2023, doi: 10.3390/electronics12040929.