

"NeuroGenesis: Agent That Learns to Perform Multiple Household Tasks"

Aparna Mote¹, Sumit Ingle², Rutuja Varunkar³, Atharva Gadade⁴

¹Aparna Mote Computer & Zeal College Of Engineering and Research,Pune

²Sumit Ingle Computer & Zeal College Of Engineering and Research,Pune

³Rutuja Varunkar Computer & Zeal College Of Engineering and Research,Pune

⁴Atharva Gadade Computer & Zeal College Of Engineering and Research,Pune

Abstract - The project "**NeuroGenesis: An Intelligent Agent for Learning and Executing Diverse Household Tasks**" aims to overcome the shortcomings of traditional smart home systems that primarily depend on fixed schedules, predefined rules, and simple automation mechanisms. Such systems often lack flexibility and are unable to respond effectively to changing household conditions and dynamic environments. This project introduces a simulation-based, AI-powered household agent designed to perform a variety of domestic tasks while continuously learning from its surroundings. By analyzing environmental changes in real time, the agent can adjust its actions, make independent decisions, and improve overall operational efficiency and user convenience.

In contrast to conventional smart home solutions that rely heavily on scripted behaviors or direct user commands, NeuroGenesis incorporates advanced **Reinforcement Learning (RL)** techniques and adaptive decision-making capabilities. These technologies enable the agent to understand complex situations, optimize its behavior through experience, and manage multiple household activities autonomously. Built on a scalable simulation environment, the project showcases the potential of intelligent agents to move beyond rigid automation toward highly adaptive and context-aware home assistance. The proposed system represents a significant advancement toward next-generation smart living environments that are resilient, personalized, and capable of intelligent self-improvement.

Keywords: Smart Home Systems, Adaptive AI, Intelligent Household Agent, Reinforcement Learning, Dynamic Learning, Simulation Environment, Autonomous Decision-Making, Task Generalization, Artificial Intelligence, Context-Aware Computing, Multi-Task Learning, General-Purpose AI, Smart Environments.

1. INTRODUCTION

The continuous evolution of smart home technology has significantly improved the way people interact with and manage their living spaces. Most modern smart home systems rely on predefined schedules, voice-controlled commands, or rule-based automation to perform household functions. Although these methods work well in predictable situations, they often struggle to respond

effectively to unexpected events and changing environmental conditions. As a result, conventional smart home solutions lack the adaptability and intelligence necessary to operate efficiently in complex, real-world settings.

Artificial Intelligence (AI) has emerged as a promising solution for enhancing the capabilities of smart home systems by enabling agents to learn, reason, and make decisions autonomously. Despite recent advancements, many AI-based household systems still encounter difficulties in areas such as handling multiple tasks simultaneously, understanding contextual information, and adapting to new situations in real time [1]. This has created a growing need for intelligent agents that can continuously learn from experience, modify their behavior based on environmental changes, and independently perform a wide range of household activities.

To address these limitations, the **NeuroGenesis** project introduces a simulation-driven household agent powered by **Reinforcement Learning (RL)**. Rather than relying on fixed automation rules, the proposed agent improves its performance through continuous interaction with its environment and adaptive decision-making processes. Drawing inspiration from both model-based and model-free reinforcement learning approaches used in robotic control and manipulation [2], the system combines learned behaviors with structured knowledge to achieve greater flexibility and task generalization. The proposed framework demonstrates a transition from traditional automation toward intelligent, adaptive, and context-sensitive home environments capable of responding effectively to dynamic real-world conditions [3]

2. LITERATURE REVIEW

Recent research in intelligent robotics has shown that Reinforcement Learning (RL) enables autonomous agents to learn optimal actions through continuous interaction with the environment. Unlike traditional rule-based systems, RL allows robots to improve their performance by maximizing cumulative rewards over multiple training episodes. The use of simulation platforms such as PyBullet has further accelerated the development of robotic learning by providing realistic and safe training environments. These advancements

have established RL as a promising solution for developing adaptive household robots capable of handling dynamic situations.

Computer vision has become an essential component of modern robotic systems for understanding and interacting with surrounding environments. Deep learning models such as Convolutional Neural Networks (CNNs) and YOLO are widely used for object detection, recognition, and localization in indoor environments. These techniques enable robots to identify household objects, estimate their positions, and generate meaningful state representations for decision-making. The integration of computer vision with reinforcement learning significantly improves task accuracy and environmental awareness.

Several researchers have applied advanced reinforcement learning algorithms, including Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN), to robotic navigation and manipulation tasks. These algorithms have demonstrated excellent performance in path planning, obstacle avoidance, robotic grasping, and motion control while maintaining stable learning behavior. However, most existing approaches are designed for solving individual tasks and lack the capability to perform multiple household activities using a unified learning framework. This limitation reduces their adaptability in practical smart home environments. The proposed NeuroGenesis framework addresses these challenges by combining PyBullet simulation, computer vision, deep reinforcement learning, and robotic control into a single intelligent architecture. Unlike conventional household automation systems, the proposed agent continuously learns from experience and adapts to changing environmental conditions without requiring manual reprogramming. The system is designed to perform multiple household tasks such as navigation, object manipulation, and cleaning using a shared reinforcement learning policy. This integrated approach improves scalability, adaptability, and decision-making, making NeuroGenesis a promising solution for next-generation intelligent home assistants.

3. PROPOSED METHODOLOGY

A. System Architecture

The proposed system, NeuroGenesis, is designed as an intelligent robotic framework capable of autonomously learning and performing multiple household tasks. The architecture follows a modular design based on a Deep Reinforcement Learning (DRL) framework, enabling adaptive decision-making and continuous learning. The system consists of five interconnected layers.

A. Simulation and Environment Layer

This layer serves as the virtual environment where the intelligent agent is trained and evaluated.

- The PyBullet physics engine is utilized to simulate realistic robotic interactions, object

manipulation, and environmental dynamics within a household setting.

- A virtual household environment is created, including robotic components, furniture, household objects, and smart devices that represent the operational workspace.
- The simulation generates environmental state information such as robot joint positions, object locations, collision data, and task-related feedback.
- Based on the success and efficiency of completed actions, the environment provides a reward signal that guides the learning process of the reinforcement learning agent.

B. Perception Layer

The perception layer enables the agent to understand and interpret its surroundings.

- Environmental information is obtained through simulated sensors, including cameras and object-state data available within the PyBullet environment.
- Computer vision and deep learning techniques, such as Convolutional Neural Networks (CNNs) and object detection models, are employed to recognize and locate household objects.
- Relevant features extracted from sensor inputs are transformed into a compact state representation containing spatial and contextual information.
- This processed state vector serves as the primary input to the decision-making module.

C. Decision-Making Layer

The decision-making layer forms the intelligence core of NeuroGenesis and is responsible for selecting optimal actions.

- The agent utilizes advanced Deep Reinforcement Learning algorithms such as Proximal Policy Optimization (PPO) or Deep Q-Networks (DQN) to learn task-specific policies.
- A high-level task manager determines which household task should be executed based on the current environment state.
- The RL agent continuously evaluates possible actions and selects those that maximize long-term rewards.
- The reward mechanism is designed to encourage efficient task completion, minimize unnecessary actions, and improve overall system performance.
- Through continuous interaction with the environment, the agent learns adaptive behaviors that generalize across multiple household scenarios.

D. Control Layer

The control layer converts the agent's decisions into executable robotic actions.

- High-level actions generated by the RL agent are translated into robot movement commands.

- Classical robotic control methods, including Inverse Kinematics (IK) and motion planning techniques, are used to ensure smooth and accurate execution.
- The control module manages robot navigation, object grasping, object placement, and other household operations.
- Safety constraints and collision-avoidance mechanisms are incorporated to ensure stable operation within the simulated environment.

E. Data Management and Integration Layer

This layer supports training, evaluation, and communication between all system components.

- Interaction data, including states, actions, rewards, and task outcomes, are recorded for performance monitoring and analysis.
- Trained model parameters and checkpoints are periodically stored to support retraining and future deployment.
- Python-based APIs facilitate communication between the PyBullet simulation, reinforcement learning framework, perception models, and control modules.
- The modular design allows easy integration of additional tasks, sensors, and learning algorithms.

F. System Workflow

The proposed **NeuroGenesis** framework follows a continuous closed-loop reinforcement learning workflow that enables the robotic agent to perceive its surroundings, make intelligent decisions, execute actions, and continuously improve its performance through interaction with the environment. The overall workflow is illustrated as follows:

Step1: Environment Initialization Initially, the PyBullet simulation environment is created with a virtual household consisting of rooms, furniture, household objects, obstacles, and a robotic manipulator. The initial positions of the robot and household objects are randomized to improve the generalization capability of the learning model. Depending on the experiment, a household task such as object pickup, object placement, navigation, or cleaning is assigned to the robot.

Step 2: Sensor Data Acquisition The robot continuously gathers environmental information using virtual RGB cameras, depth sensors, and positional sensors available in the simulation. Additional information including object coordinates, robot joint angles, collision status, and distance measurements is also collected. These observations together represent the current state of the environment.

Step3: Perception and Feature Extraction The perception module processes the acquired sensor data using computer vision techniques. CNN/YOLO-based object detection identifies household objects and estimates their locations. Image preprocessing, feature extraction, and state encoding are then performed to convert raw sensory information into meaningful numerical features. These extracted features are combined with the robot's internal state to construct the final state representation.

Step4: State Representation The processed observations are transformed into a compact state vector suitable for reinforcement learning. The state representation includes the robot's position and orientation, joint configurations, target object locations, goal position, obstacle locations, distance to the target, and task completion status. This state vector serves as the input to the reinforcement learning agent.

Step 5: Decision Making Using Reinforcement Learning

The reinforcement learning agent receives the current state vector and processes it using a trained PPO or DQN policy network. The policy evaluates all possible actions and selects the action that maximizes the expected cumulative reward while minimizing task completion time and energy consumption. During training, exploration and exploitation strategies are balanced to discover optimal policies.

Step 6: Action Selection Based on the policy output, the robot selects an appropriate action such as moving forward, turning left or right, navigating toward the target object, picking up an object, placing an object, opening or closing a door, avoiding obstacles, or returning to its home position. The selected action is then forwarded to the robot control module.

Step 7: Motion Planning and Robot Control The control module converts the selected high-level action into executable robotic commands. Inverse kinematics is employed to compute the required joint trajectories for the robotic manipulator, while motion planning algorithms generate smooth and collision-free paths. The robot then executes these movements within the PyBullet simulation.

Step 8: Environment Interaction After the action is executed, the robot interacts with the virtual household environment. Object positions, robot location, and environmental conditions are updated accordingly. Dynamic interactions including navigation, object manipulation, and collision handling are simulated in real time to reflect realistic household scenarios.

Step 9: Reward Computation The environment evaluates the outcome of the executed action and computes a numerical reward. Positive rewards are assigned for successful task completion, reduced distance to the target, efficient path planning, collision avoidance, and energy-efficient behavior, whereas penalties are imposed for collisions, inefficient movements, or task failures. This reward signal guides the learning process toward optimal behavior.

Step 10: Policy Update The current state, selected action, received reward, and next state are stored as an experience tuple. During training, the reinforcement learning algorithm updates the neural network parameters using these experiences to maximize the cumulative reward over multiple training episodes.

Step 11: Continuous Learning Cycle The updated policy repeatedly interacts with the simulation environment over thousands of episodes. Through continuous trial-and-error learning, the robotic agent gradually improves its navigation, manipulation, and decision-making capabilities until the policy converges to an optimal solution.

Step 12: Task Completion and Performance Evaluation

Upon successful completion of a household task, several performance metrics including task success rate, cumulative episode reward, learning curve, average completion time, collision rate, energy consumption, and path efficiency are recorded. These metrics are used to evaluate the effectiveness and robustness of the proposed NeuroGenesis framework.

Step 13: Generalization to Multiple Household Tasks

After training, the learned policy is evaluated in previously unseen household environments containing different object arrangements and layouts. The trained agent demonstrates its ability to generalize across multiple household tasks, including object fetching, indoor navigation, item organization, cleaning assistance, and object delivery, without requiring manual reprogramming or additional task-specific training.

4. SYSTEM ARCHITECTURE

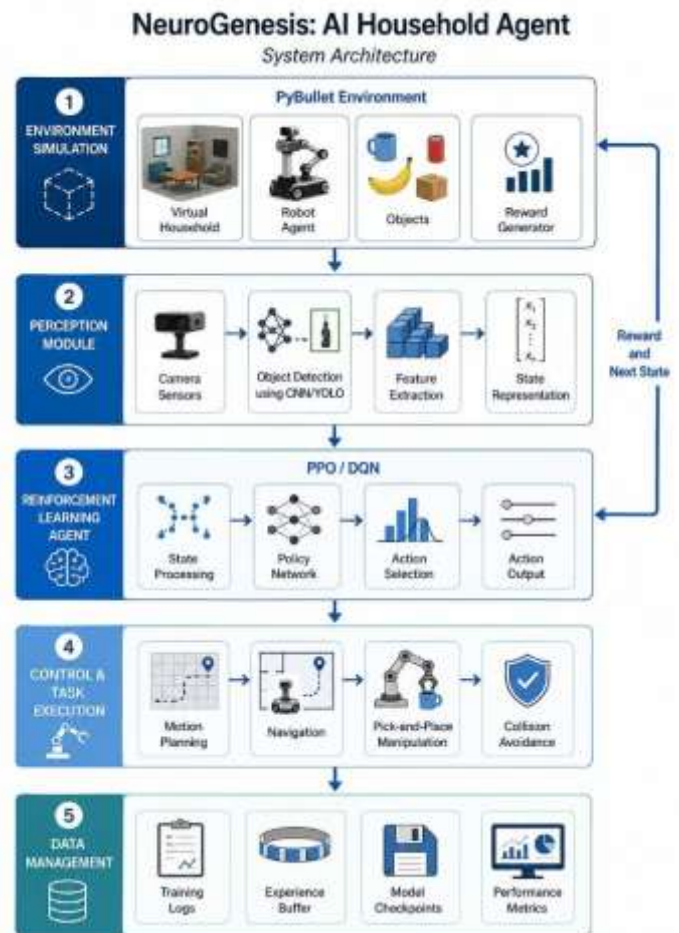


Fig-1: System Architecture Diagram

5. IMPLEMENTATION

The proposed NeuroGenesis system was implemented using the PyBullet physics simulation environment to create a realistic virtual household for training and evaluating the intelligent robotic agent. The environment contains household furniture, tables, cabinets, and multiple objects that serve as task targets for the robot. A robotic manipulator equipped with a gripper performs household operations such as navigation, object detection, grasping, and object placement. The perception module continuously acquires environmental information through simulated sensors and processes object locations to generate the current state representation. Based on this state, the reinforcement learning agent, trained using the Proximal Policy Optimization (PPO) algorithm, selects optimal actions to maximize cumulative rewards while minimizing unnecessary movements. The control module converts these high-level decisions into joint trajectories using inverse kinematics, enabling smooth and collision-free robotic motion. During execution, the robot successfully approaches target objects, grasps them, and transports them to designated locations while avoiding surrounding obstacles. The environment dynamically updates after every action and provides reward feedback, allowing the policy to improve continuously through

repeated interactions. The implementation demonstrates the effectiveness of combining PyBullet simulation, computer vision, reinforcement learning, and robotic control to develop an adaptive household assistant capable of performing multiple domestic tasks autonomously.

Then insert your screenshots immediately after this section with captions such as:

- Fig.2 Environment Initialization
- Fig.3 Agent Navigation
- Fig.4 Object Detection
- Fig.5 Object Manipulation
- Fig.6 Task Execution
- Fig.7 Task Completion
- Fig.8 PyBullet Household Environment



Fig.4 Object Detection



Fig.2 Environment Initialization



Fig.5 Object Manipulation

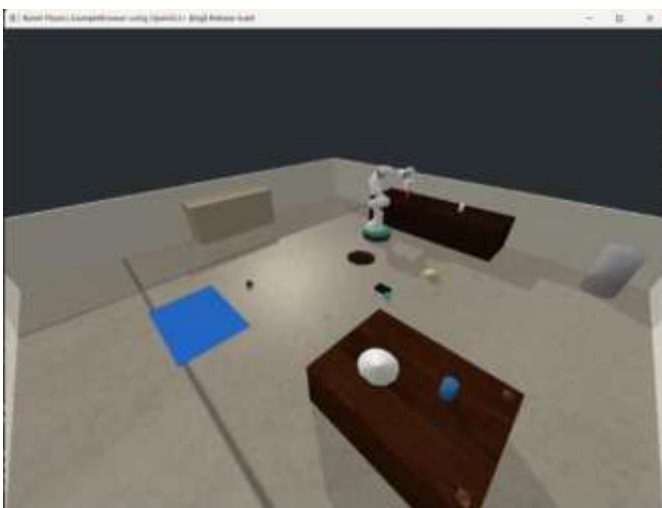


Fig.3 Agent Navigation



Fig.6 Task Execution



Fig.7 Task Completion

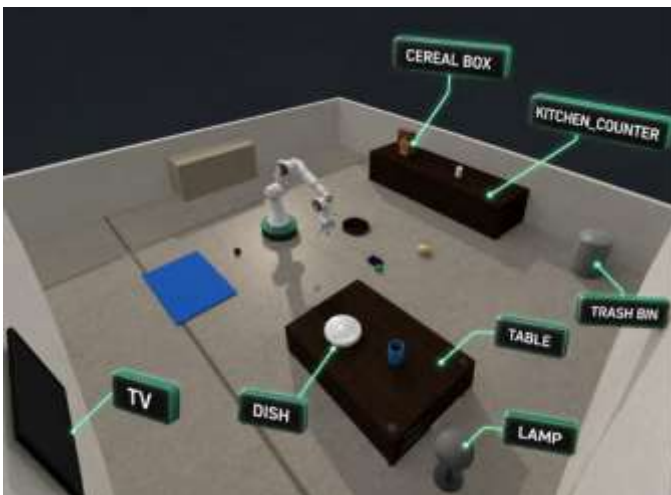


Fig.8 PyBullet Household Environment

6. RESULTS / DISCUSSION

The **NeuroGenesis** agent was tested within a simulated household environment that mimics everyday domestic situations and challenges. The primary objective of the evaluation was to assess the agent's ability to complete tasks accurately, operate efficiently, and adapt to changing environmental conditions without human intervention.

Example Tasks Performed

- Room cleaning and organization
- Switching off unused electrical appliances
- Picking, moving, and placing household objects in designated locations

Discussion

Experimental observations show that the **NeuroGenesis** agent is capable of learning effective strategies and

adjusting its behavior when faced with modifications in the environment, such as changes in object positions or room layouts. The agent successfully handled multiple household tasks without requiring manual rule updates or task-specific programming.

Training performance, measured through cumulative reward progression over episodes, revealed a consistent increase in learning efficiency and task success rates. As training progressed, the agent demonstrated improved decision-making, reduced unnecessary actions, and faster task completion. Although occasional inaccuracies were observed during more complex interactions, overall performance remained stable and reliable across different test scenarios.

The findings suggest that reinforcement learning can effectively support the development of adaptive household agents capable of operating in dynamic environments. Future enhancements may include incorporating advanced sensing technologies, refining reward design to encourage more optimal behavior, and expanding the simulation environment with additional tasks and complexities. These improvements are expected to further strengthen the agent's adaptability, robustness, and real-world applicability.

7. LIMITATIONS

1. Simulation-Based Evaluation: The proposed system is developed and evaluated entirely in the PyBullet simulation environment. Although the simulator provides realistic physics, it cannot fully capture the uncertainties and complexities of real-world household environments.

2. Limited Task Diversity: The current implementation focuses on a predefined set of household tasks such as navigation, object pickup, object placement, and cleaning. More complex activities involving human interaction, long-term planning, or multi-step reasoning are not considered.

3. High Computational Cost: Training deep reinforcement learning models such as PPO and DQN requires a large number of training episodes and significant computational resources, resulting in long training times.

4. Dependence on Reward Function Design: The learning performance heavily depends on the design of the reward function. Poorly designed rewards may lead to suboptimal policies or unintended robot behaviors.

5. Limited Generalization: Although the agent can adapt to different simulated scenarios, its performance may

decrease when introduced to completely unseen environments, object types, or household layouts.

6.Perception Constraints: Object detection and state estimation rely on simulated sensor data and ideal environmental conditions. Real-world challenges such as sensor noise, lighting variations, occlusions, and dynamic obstacles are not fully addressed.

7.Lack of Multi-Agent Collaboration: The proposed framework considers only a single intelligent household agent. Collaborative task execution involving multiple robots or interaction with human users is beyond the scope of this work.

8.Sim-to-Real Transfer Challenges: Policies learned in simulation may not directly transfer to physical robots due to differences in hardware dynamics, sensor inaccuracies, and environmental uncertainties. Additional domain adaptation and sim-to-real transfer techniques would be required for practical deployment.

8. CONCLUSION

This research introduces **NeuroGenesis**, an intelligent household agent powered by **Reinforcement Learning (RL)** that can learn and execute a variety of domestic tasks through continuous adaptation and autonomous decision-making. The proposed system demonstrates how learning-based intelligence can overcome the limitations of conventional smart home automation by enabling more flexible, responsive, and context-aware behavior. Experimental results obtained from the simulation environment confirm the effectiveness of the approach in achieving reliable task execution, adaptability to changing conditions, and scalability for handling multiple household activities.

Looking ahead, future work will focus on extending the system beyond simulation into real-world household environments. Additional research directions include enhancing collaboration among multiple intelligent agents, improving task coordination in complex scenarios, and incorporating **Explainable Artificial Intelligence (XAI)** techniques to provide greater transparency and user understanding of the agent's decisions. These advancements will contribute to building more trustworthy, efficient, and user-centric smart home systems, as highlighted in previous studies [3].

9.ACKNOWLEDGEMENT

The authors sincerely express their gratitude to their project supervisor for continuous encouragement, expert guidance, and valuable support throughout the development of this work. They also acknowledge the

contributions of the open-source community, particularly the developers of frameworks and tools such as **TensorFlow**, **PyTorch**, and **PyBullet**, which provided essential resources for simulation, machine learning, and system implementation. Special thanks are extended to academic colleagues, faculty members, and the broader research community whose discussions, feedback, and shared knowledge significantly contributed to the successful completion of this project

10. REFERENCES

- [1] N. K. K. Nguyen, A. T. Mang, and Q. Nguyen, "A Lightweight PyBullet-Based Framework for Fast Reinforcement Learning Prototyping on 6-DOF Robotic Arms," *Journal of Mechatronics and Robotics*, vol. 9, no. 1, pp. 35–39, 2025.
- [2] Q. Zhang, G. Han, J. Sun, W. Zhao, C. Sun, J. Cao, J. Wang, Y. Guo, and R. Xu, "Distillation-PPO: A Novel Two-Stage Reinforcement Learning Framework for Humanoid Robot Perceptive Locomotion," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2025, pp. 2916–2922.
- [3] R. Ozalp, A. Ucar, and C. Guzelis, "Advancements in Deep Reinforcement Learning and Inverse Reinforcement Learning for Robotic Manipulation: Toward Trustworthy, Interpretable, and Explainable Artificial Intelligence," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 12, 2024.
- [4] Z. Wu, Y. Li, W. Zhan, C. Liu, Y.-H. Liu, and M. Tomizuka, "Efficient Reinforcement Learning of Task Planners for Robotic Palletization Through Iterative Action Masking Learning," *IEEE Robot. Autom. Lett.*, Nov. 2024, pp. 1–8, doi: 10.1109/LRA.2024.3440731, 2024.
- [5] M. S. Nazeer, C. Laschi, and E. Falotico, "RL-Based Adaptive Controller for High Precision Reaching in a Soft Robot Arm," *IEEE Trans. Robot.*, vol. 40, 2024.
- [6] C. Chronis, G. Anagnostopoulos, E. Politi, G. Dimitrakopoulos, and I. Varlamis, "Dynamic Navigation in Unconstrained Environments Using Reinforcement Learning Algorithms," *IEEE Access*, vol. 11, 2023.
- [7] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, and X. Xu, "Deep Reinforcement Learning: A Survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, 2022.
- [8] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, and S. Yogamani, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, 2022.

[9] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep Reinforcement Learning for Robotic Manipulation: A Survey,” *Foundations and Trends in Robotics*, vol. 10, no. 3–4, pp. 143–379, 2022.

[10] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa, and M. Hayashibe, “A Survey of Sim-to-Real Transfer Techniques Applied to Reinforcement Learning for Bioinspired Robots,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, 2021.