International Research Journal of Multidisciplinary Scope (IRJMS), 2024; 5(1): 336-354

Original Article | ISSN (O): 2582-631X DOI: 10.47857/irjms.2024.v05i01.0213

Reinforcement Learning-Based Collision Avoidance of Kinova Gen3 Robot for Ball Balancing Task

Aishwarya Chandrakant Kadam, Sudhir Madhav Patil*

Department of Manufacturing Engineering and Industrial Management, COEP Technological University (COEP Tech), Chhatrapati Shivajinagar, Pune- 411005, Maharashtra, India. *Corresponding Author's Email: smp.prod@coeptech.ac.in

Abstract

Reinforcement learning (RL) has arisen as a formidable approach to empower robots to acquire intricate skills by engaging with their surroundings. This study presents a novel application of RL techniques to address the challenging problem of collision avoidance in the context of a Kinova Gen3 robotic arm tasked with ball balancing. The goal of the robot is to maintain dynamic equilibrium of a ball on its end effector while navigating a constrained workspace and avoiding collision with obstacles. Modern RL algorithms are used in this approach to enable the robot to learn collision-free control strategies through data-driven learning. Specifically, this paper employs RL, combining actor-critic architecture with advanced exploration strategies to learn optimal collision avoidance behaviors efficiently. The actor component learns a policy that determines the robot's actions to maintain the ball's balance, while the critic component estimates the value function to provide valuable feedback for policy improvement. To facilitate effective learning, a realistic simulation environment was designed. The Kinova Gen3 robotic arm interacts with simulation environment to collect large amounts of data, which are then utilized to train and refine the RL-based collision avoidance policy. The effectiveness and adaptability of the suggested RL-based collision avoidance technique is validated by experimental simulation results. The Kinova Gen3 robotic arm successfully learns to perform the ball balancing task while maintaining safety through intelligent collision avoidance strategies. By showing how RL can be used to help robots accomplish complex tasks like dynamic equilibrium and real-time obstacle avoidance, this research advances robotic control methodologies.

Keywords: Ball balancing, Collision avoidance, Kinova Gen3 robot, Reinforcement learning, Robotics simulation, Soft actor-critic.

Introduction

Recent advancements in the field of robotics have demonstrated the possibility for robots to carry out difficult tasks on their own. One crucial aspect of this progress is the development of intelligent control strategies that enable robots to operate safely and efficiently in dynamic environments (1, 2). A key challenge in achieving this is of collision avoidance, which involves navigating an environment while preventing unintended physical contact with obstacles or other entities (3, 4). The advanced robotic platform known as the Kinova Gen3 robot has several degrees of freedom and adaptable end effectors, making it ideal for a variety of applications, including pick-and-place, assembly, and manipulation operations (5). Addressing collision avoidance is particularly critical for scenarios where precision and dexterity are essential, such as in the context of a Kinova Gen3 robotic arm engaged in a ball-balancing task. In the ball balancing task, the robot's objective is to maintain an equilibrium of a ball on its end effector while operating within a confined workspace. This task inherently requires precise control to counteract the ball's movements and ensure its stability.

Robots are now being trained to learn complicated tasks and adapt to a variety of contexts using reinforcement learning (RL), a revolutionary technique. Robots can learn from experience through trial and error using RL, which is different from conventional rule-based approaches. This allows robots to gradually improve their behavior based on the results of their activities (6). This paper explores the integration of RL approaches for the Kinova Gen3 robotic arm to perform the ball-balancing task more effectively. The major objective of this research is to create an intelligent collision

(Received 05th November 2023; Accepted 09th January 2024; Published 30th January 2024)

avoidance technique that enables the Kinova Gen3 robotic arm to navigate an environment filled with obstacles while maintaining a safe and stable ball balance. The objective is achieved by leveraging the principles of RL and seeking to imbue the robot with the ability to autonomously perceive its surroundings, evaluate potential collision risks, and make real-time decisions to prevent unwanted contact. By effectively addressing collision avoidance, the paper aims to augment the robot's versatility, reliability, and adaptability in dynamic environments.

The structure of the paper is as follows: Section: Literature review, covers the survey of the literature. Sub-section: Reinforcement learning, discusses related work on the use of RL in robotics and sub-section: Soft actor-critic algorithm, discusses related work on soft actor-critic (SAC) algorithm used in obstacle avoidance for robot environment. Section: Materials and methods, includes sub-section: Kinova Gen3 robot, which contains detailed description of Kinova Gen3 robot. Sub-section: Integration: Kinova Gen3 robot and collision avoidance system presents steps for integration between Kinova Gen3 robotic arm and collision avoidance system. Sub-section: Experiment simulation mentions the details about the experimental simulation performed for collision avoidance. Section: Results, presents outcomes of simulation. Section: Discussion, opens with the mention of limitations of the current study, further contains detailed discussion surrounding the results received from experimental simulation and ends with concluding remarks with some final thoughts and recommendations for the future research.

In summary, this work is a major step toward applying RL techniques to give, the Kinova Gen3 robotic arm, intelligent collision avoidance capabilities for the ball balancing task. The paper's solution to this basic problem advances the objective of improving the performance of robots functioning in dynamic and unstructured environments, opening the door to the development of robotic systems that are safer, more effective, and more adaptable.

Literature review

Reinforcement learning

RL, a branch of machine learning (ML), focuses on instructing agents (such as software or robots) about how to make decisions based on input from their environment. The agent learns to draw conclusions that maximize a reward signal in order to make better decisions (7, 8). Through the use of RL algorithms, a robot can adjust and learn from its interactions with its surroundings. This is especially helpful for tasks requiring the robot to maneuver through intricate or dynamic environments with shifting obstacles (9). RL models have the ability to apply their training data's knowledge to novel and unfamiliar scenarios. This implies that the robot doesn't need a lot of reprogramming to handle a variety of obstacle configurations and adapt to novel situations. Robots can learn by trial and error, which is frequently the most efficient way to find the best routes around obstacles, thanks to RL (10, 11). In order to create efficient obstacle avoidance policies, it can experiment with various actions and track the results. The robot can gradually enhance its obstacle avoidance techniques by using RL. It can improve its policies and become more adept at avoiding obstacles as it engages with the environment and gathers feedback (12). Many tasks involving robot manipulation involve high-dimensional state spaces, for which it can be difficult to apply conventional rule-based or hard-coded methods. Such intricate state spaces can be handled by RL, and it can acquire useful policies in these conditions (13). When a robot must navigate unexpected obstacles or changes in the surroundings, or when it must perform tasks in unstructured and dynamic environments where obstacles are not known with precision beforehand, RL is a good fit. Creating obstacle avoidance behaviors in traditional robotics often requires a great deal of human programming. Because RL allows the robot to learn obstacle avoidance techniques on its own, it can drastically reduce the need for explicit programming (14).

RL stands as a promising frontier in foreseeing and preventing collisions across various domains. By leveraging iterative learning through interactions with simulated or real environments, RL agents can discern complex patterns and anticipate collision scenarios. These models learn from trial and error, continuously refining strategies to navigate environments and avoid potential collisions. RL also allows for the creation of adaptive collision avoidance strategies that can evolve in response to changing scenarios, offering potential applications in autonomous vehicles, robotics, aviation, and other safety-critical domains. The ability to proactively predict and avert collisions showcases the potential of RL in enhancing safety and efficiency across a spectrum of industries. There are different types of algorithms present in RL for performing manipulation tasks. Figure 1 illustrates the detailed classification of RL algorithms.

Soft actor-critic algorithm

Model-free RL is a widely used learning method. SAC is a type of policy gradient technique. It is intended to support exploration in a steady way and manage continuous action spaces effectively. SAC introduces entropy regularization to the policy optimization process, which leads to a more exploratory policy and better performance in complex environments (15). When it comes to sample efficiency, SAC is well-known to be superior to certain other RL algorithms. Given that gathering data can be expensive and time-consuming, this can be essential for training in real-world scenarios. In tasks involving physical systems, such as robot manipulation, efficient learning is particularly crucial (16). SAC uses methods to promote stable and strong policies, such as entropy regularization. This can assist in teaching the robot policies that minimize the chance of collisions by avoiding obstacles and doing so in a smooth and controlled manner (17). Continuous control in high-dimensional action spaces is necessary for many robot manipulation tasks. SAC can learn policies that give it fine-grained control over robot movements, making it an excellent choice for tasks requiring continuous action spaces (18). Stochastic policy, which SAC employs, naturally includes exploration. This is essential for the robot to learn efficient obstacle avoidance techniques because it must experiment with different actions to find safe routes around obstacles (19). SAC is capable of learning from previously gathered data because it supports off-policy learning. For tasks where it is unsafe or impractical to collect data in real-time, this is advantageous. The algorithm can enhance obstacle avoidance by optimizing previous experiences (13, 20). Dense rewards are hard to come by for RL in many real-world settings. When learning from sparse reward signals, as is frequently the case in obstacle avoidance tasks, SAC can handle it well (21). Compared to some other RL algorithms, SAC usually requires fewer hyper parameters to adjust, making it more user-friendly for individuals who are not RL experts (22).

There are some key concepts of the SAC algorithm. Entropy regularization: Entropy regularization in SAC maximizes policy entropy and expected cumulative reward, encouraging exploratory behavior and better policy discovery in complex, uncertain environments (23). Off-policy learning: SAC is an off-policy algorithm that can learn from data from any policy, enhancing efficiency by reusing past experiences from different policies (24). Dual Q-functions: The twin-critic setup in SAC employs two Q-functions to estimate the action-value function, thereby mitigating overestimation bias and ensuring more stable value function estimates (25).

Materials and methods

RL-based manipulation in a confined environment is the application of RL techniques to help robotic systems or agents perform manipulation tasks under specific constraints. Constrained environments can include scenarios with limited workspace, safety constraints, or other restrictions that the agent must consider while performing the manipulation tasks. Applying RL-based manipulation in constrained environments requires a thoughtful design of the task, observation, action spaces, reward function, and safety considerations. It also involves extensive testing and evaluation to ensure that the agent can perform the manipulation task safely and effectively within the specified constraints. This section presents the details of Kinova Gen3 robotic arm, integration between Kinova Gen3 robotic arm and collision avoidance system. This section also talks about safety and effectiveness of the proposed RL based approach for collision avoidance. It explains about how RL based collision avoidance system improves security and effectiveness of robot’s ball balancing task. It also demonstrates the use of a SAC algorithm and describes the experimental simulation of the Kinova Gen3 robotic arm for ball balancing task by collision avoidance.

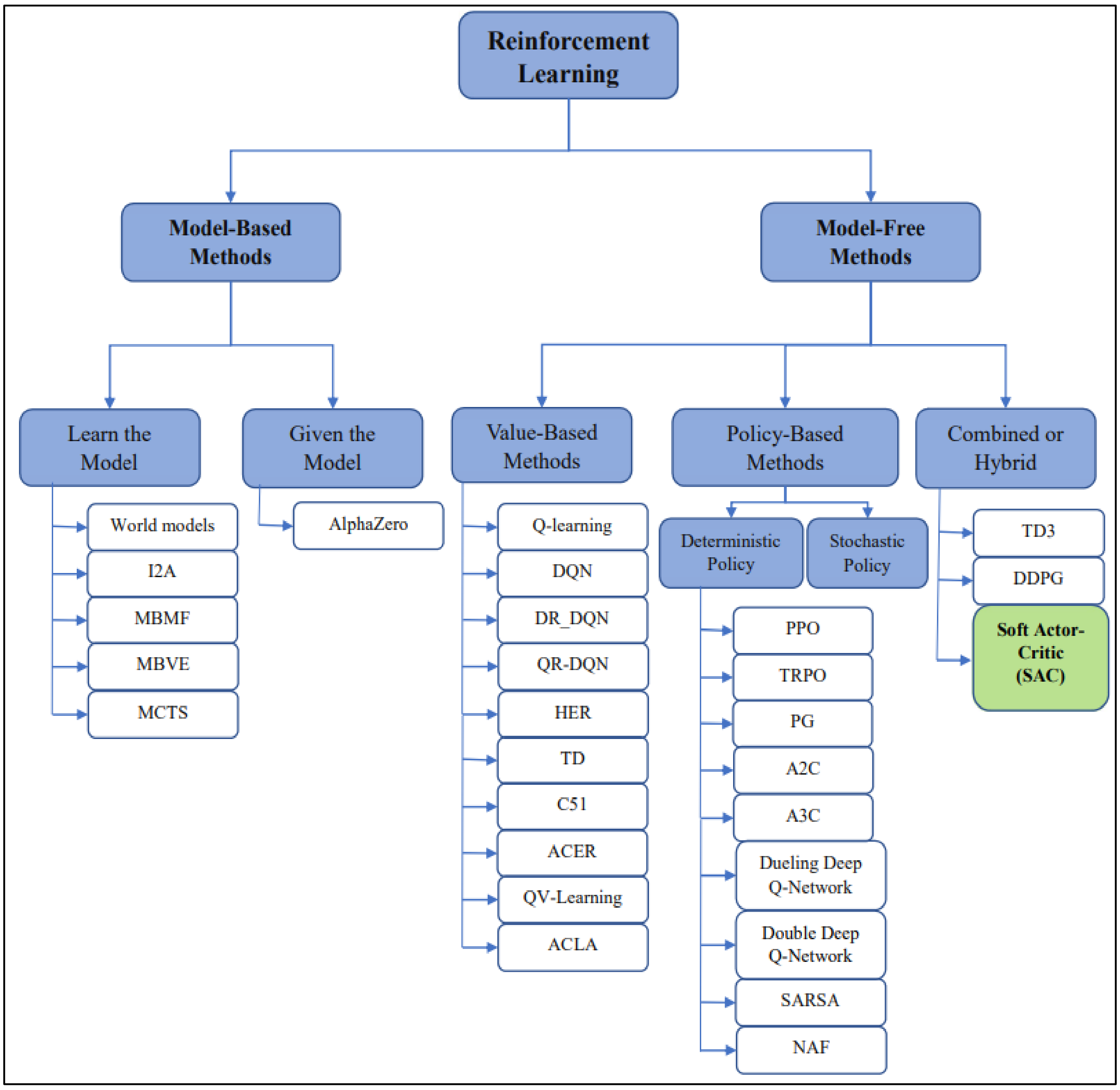


Figure 1: Classification of RL algorithms

Kinova Gen3 robot

The company Kinova Robotics, which is renowned for producing creative and adaptable robotic solutions, created the 7-degree-of-freedom (7-DOF) Kinova Gen3 robotic manipulator. With a focus on enhancing human-robot collaboration and improving the quality of life for people with mobility limitations, the Gen3 robot is intended for a range of applications, including research, manufacturing, and assistive technology. Figure 2 depicts the primary parts of the Kinova Gen3 system. Its design is compact and it weighs incredibly little. It features user-friendly programming and control interfaces, customizable end-effectors and accessories for task-specific adaptability, force and torque sensors for safety and feedback, and compatibility with various simulation programs like MATLAB/Simulink and robot operating system (ROS). These elements emphasize accuracy, modularity, and user-friendliness, and together they make the Kinova Gen3 robot a flexible and collaborative platform for a range of applications (26).

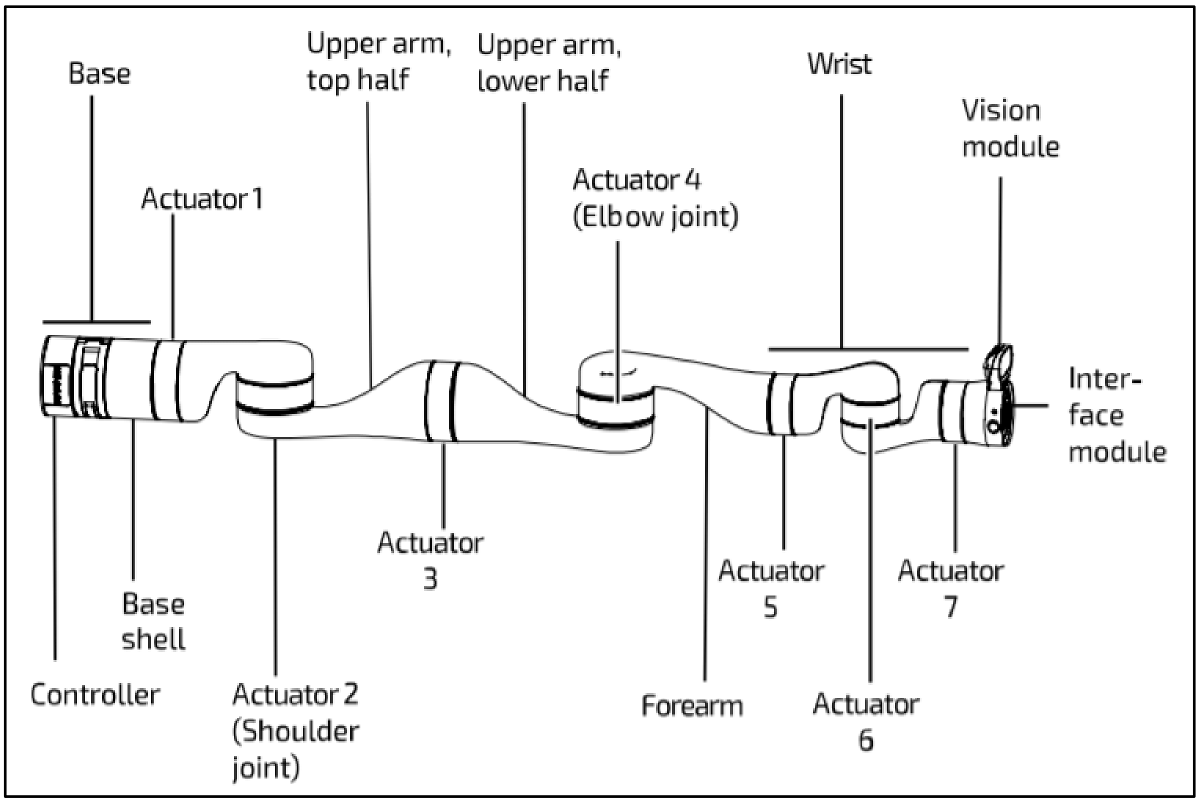


Figure 2: Primary parts of Kinova Gen3 robot "adapted from Kinova Robotics (26)"

There are two main ways that the Kinova Gen3 robotic arm can be controlled. The first is a high-level application programming interface (API) that makes use of the Kinova Gen3 controller library's angular or cartesian control options. Through access to specific actuators, the second approach, the low-level API, enables users to directly control the robotic arm. This method is perfect for users who need more sophisticated control capabilities because it offers greater flexibility and faster feedback (27).

For the Kinova Gen3 robot's obstacle avoidance, therefore, conventional rule-based programming techniques are either too limited or too time-consuming. This would justify investigating RL methods, like SAC, as a substitute for automatically picking up obstacle avoidance techniques. While the Kinova Gen3 robotic arm may be adept at avoiding obstacles in controlled settings, it finds it difficult to apply these abilities to unfamiliar or novel situations. The goal of this work is to create SAC-based techniques that enhance generalization. One area of research that still needs to be addressed is how to train real Kinova Gen3 robots using RL algorithms such as SAC with less data and sample complexity. Closing this gap makes training more effective and realistic in real-world environments. A major concern is making sure the Kinova Gen3 robot is safe while it uses RL to learn obstacle avoidance techniques. Developing methods for integrating safety and risk reduction during training is one area of unmet research need.

Integration: Kinova Gen3 robot and collision avoidance system

Integrating a collision avoidance system with the Kinova Gen3 robot's ball-balancing task involves ensuring the safety of the robot, the environment, and any interacting objects or humans. Following are key points to consider for the integration:

Sensors: Implement a combination of sensors such as cameras, LiDAR, and proximity sensors to detect the surroundings. Utilize force/torque sensors on the robot's joints to sense unexpected contact with objects.

Collision detection: Develop algorithms for real-time collision detection based on sensor data. Implement methods to distinguish between expected contacts (with the ball) and unexpected collisions.

Collision response: Design a collision response strategy to ensure safe robot behavior. Integrate a control algorithm to stop or adjust the robot's motion upon detecting a collision.

Trajectory planning: Incorporate collision-aware trajectory planning to generate robot paths that avoid obstacles. Consider dynamic replanning to adapt to changes in the environment during task execution.

Obstacle/Human-robot interaction: Implement safety measures to detect and respond to the presence of obstacles/human in the robot's workspace. Integrate a obstacle/human-aware motion planning system to avoid collisions with obstacle/humans.

Safe zones: Define safe zones where the robot can operate without collision concerns. Implement logic to guide the robot back to a safe state if it deviates from the predefined safe zones.

Emergency stop mechanism: Integrate an emergency stop system that halts the robot's motion in critical situations. Ensure quick and reliable activation of the emergency stop mechanism.

System monitoring: Implement continuous monitoring of the collision avoidance system's performance. Provide feedback and alerts to the operator in case of any issues or limitations in the collision avoidance system.

Integration with control system algorithm(s): Integrate collision avoidance functionalities seamlessly with the Kinova Gen3 robot's control system. Ensure that the collision avoidance system does not compromise the performance of the ball-balancing task.

Testing and validation: Conduct thorough testing in simulation environments and real-world scenarios. Validate the collision avoidance system's effectiveness in preventing collisions while maintaining the efficiency of the ball-balancing task.

Documentation and training: Document the integration process, including parameters, algorithms, and safety considerations. Provide training for operators on the proper use and limitations of the collision avoidance system.

Safety, performance, and effectiveness

Integrating RL based collision avoidance system with the Kinova Gen3 robot's ball-balancing task holds significant promise in enhancing the robot's performance and safety. This integration empowers the robot to foresee potential collisions with objects or obstacles in its environment and dynamically adjust its movements to prevent these collisions. It helps robot to maintain its focus on balancing the ball while concurrently analyzing its surroundings and autonomously modifying its movements to avoid collisions with static or dynamic obstacle. This capability not only enhances the safety of the robot's operation but also ensures the continuity and efficiency of the ball-balancing task, minimizing interruptions caused by potential collisions.

Moreover, the RL based collision avoidance system with the Kinova Gen3 robot's task can serve as a paradigm for safer human-robot interactions. It showcases the potential for collaborative environments where robots work alongside humans, mitigating the risk of accidents and collisions in shared spaces. This RL integration represents a significant stride in advancing both the performance and safety aspects of robotic tasks, setting a precedent for future applications of collision avoidance strategies in various robotics domains. Additionally, the adaptive nature of RL enables the robot to learn from its experiences and improve its collision avoidance strategies over time. By continuously updating its decision-making processes based on real-time feedback, the system becomes more adept at identifying and avoiding potential collisions in various environments, thereby further enhancing the security and effectiveness of the robot's ball balancing task in diverse and dynamic settings.

Experiment simulation

The MATLAB/Simulink simulation uses a powerful 7-DOF manipulator called the Kinova Gen3 robot. The key goal of the robot is to maintain a ping pong ball balanced in the exact center of a flat surface (plate) that the robot's gripper is firmly grasping. During this task, the robot must demonstrate exceptional control and dexterity to avoid any collisions with potential obstacles present in its workspace. The ultimate goal is to showcase the robot's ability to perform delicate and precise tasks while ensuring the safety of the surrounding environment.

Outline of steps

The further part outlines the steps involved in achieving the experimental simulation setup.

Perception and sensing: The robot arm needs to be equipped with sensors like cameras or depth sensors to perceive the position and orientation of the ping pong ball and the flat surface. Algorithms for object detection and tracking can be used to determine where the ball is in the workspace of the robot. The flat surface's position and orientation must also be known by the robot, and they can either be assumed or calibrated prior to the task.

Planning and control: Once the robot has the necessary information about the ball's position and the flat surface, it can plan a trajectory to approach and pick up the ball. Inverse kinematics will be used by the robot's control system to determine the joint angles necessary to move the end-effector (gripper) to the proper position and orientation for delicately picking up the ball.

Balancing the ball: Once the ball is picked up, the robot will move to the center of the flat surface while holding the ball. To balance the ball, the robot's control system will need to constantly adjust the end-effector's position and orientation based on feedback from the sensors. Vision-based feedback control algorithms can be used to keep the ball at the center of the surface and maintain its stability.

Avoiding obstacles: To prevent collisions with obstacles, the robot must have a collision detection and avoidance system. This can be achieved using proximity sensors or depth sensors to detect obstacles in the robot's vicinity. The control system should incorporate obstacle avoidance logic to adjust the robot's trajectory when it detects an obstacle in its path. In this experiment, solid brick is considered as a static obstacle and is placed with respect to the base link of the Kinova Gen3 robot. Table 1 provides the details pertaining to the dimension and frame of obstacle with respect to the robot.

Task completion: Once the robot successfully balances the ping-pong ball in the center of the level surface without colliding, the assignment is deemed complete.

Setting up RL environment in MATLAB/Simulink

Figure 3 shows the Kinova Gen3 robotic arm block diagram along with the RL environment with a specified RL agent. This agent observes the environment, and based on observations, it provides positive and negative rewards for every correct and incorrect action, respectively. After receiving a maximum reward, the agent decides whether the work is done.

The block diagram of the RL environment prepared in MATLAB/Simulink is shown in Figure 4. The 7-DOF manipulator's end-effector is connected to the plate and ball. The plate sensor and ball sensor are further attached to the blocks to have real-time data collection of ball balancing. These sensors send data as observations and actions based on which RL agent decides the rewards to be assigned. The training is said to be done once the manipulator learns the environment completely.

The block diagram of the subsystem of the environment is shown in Figure 5. The first joint is actuated using a scope signal for the continuous movement of the arm. When the joint starts to actuate due to the signal, the RL agent SAC starts balancing the ball at the center of the plate.

Table 1: Details of the dimension and frame of obstacle i.e. solid brick

Description	**Value**
Dimensions (L,W,H) in m	0.08, 0.08, 0.3
Density in kg/m^3	1000
Mass in kg	1.92

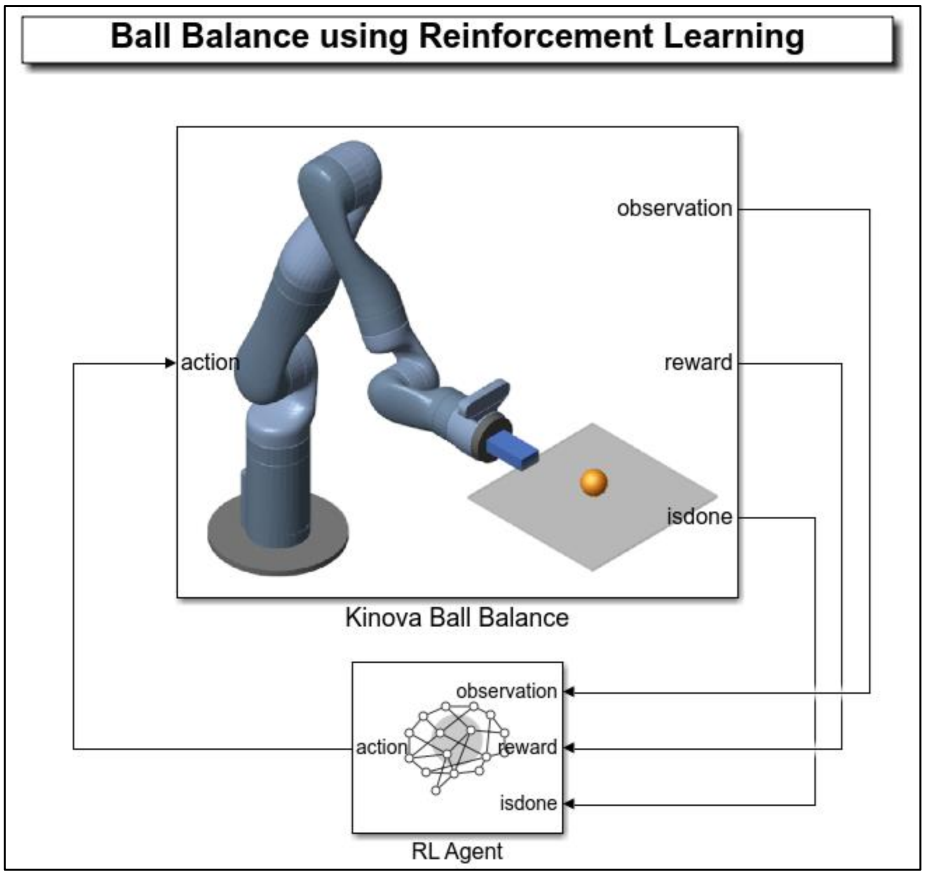


Figure 3: Kinova Gen3 robotic arm environment

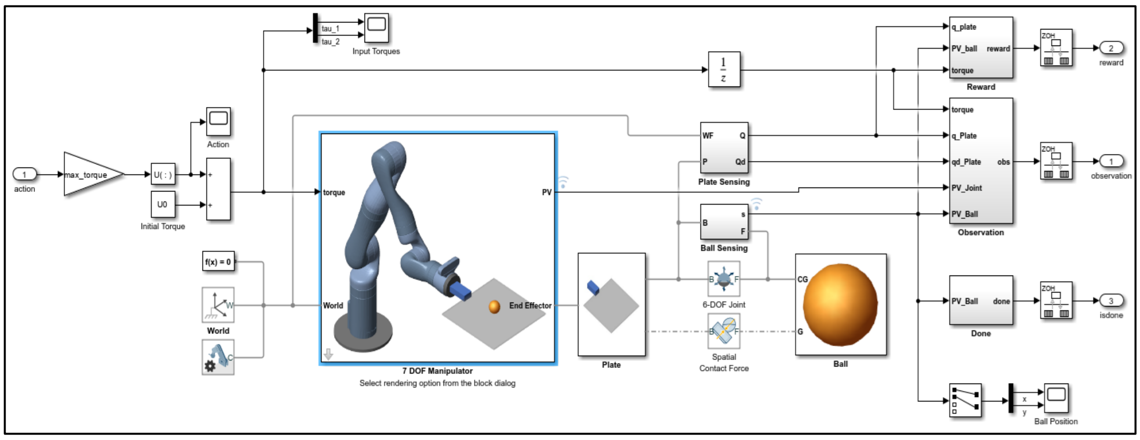


Figure 4: Block diagram of RL environment in MATLAB/Simulink

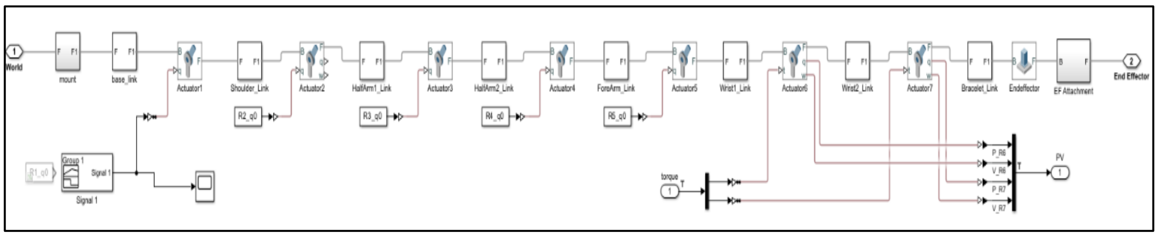


Figure 5: Block diagram of subsystem

Setting up training environment for the robotic arm in MATLAB/Simulink

To train the robotic arm to balance the ball without colliding with the obstacle using the Simscape™ Multibody™ model, further steps were followed.

Model the manipulator: (i) Create a Simscape™ Multibody™ model to represent the robotic arm (Kinova Gen3) with its multiple joints and links. (ii) Define the dynamics and kinematics of each joint in the model, including the actuated joints that receive torque signals as control inputs. The robot's control system utilizes inverse kinematics to determine the joint angles necessary to position and orient the end-effector (gripper) in the proper position for delicately picking up the ball.

Model the ball: (i) Add a separate Simscape™ Multibody™ component to represent the ping pong ball with six degrees of freedom for translational and rotational motion. (ii) Define the mass, inertia, and other physical properties of the ball in the model.

Model the plate: (i) Add another Simscape™. To represent the plate that is attached to the manipulator's end effector, use a Multibody™ component. Inverse kinematics will be used by the robot's control system to determine the joint angles necessary to move the end-effector (gripper) to the proper position and orientation for delicately picking up the ball. (ii) The plate should be constrained to move along with the end effector, meaning its position and orientation are affected by the manipulator's movements.

Implement the spatial contact force: (i) Use the Simscape™ Multibody™ Spatial Contact Force block to model the interaction between the ball and the plate when they come into contact. (ii) Configure the block to calculate and apply the contact forces and torques between the ball and the plate based on their interaction.

Training the robotic arm in MATLAB/Simulink environment

The following steps are used to train the robotic arm to balance the ball without colliding with the obstacle:

(a) Define environment
(b) Create agent
(c) Training an agent
(d) Simulate trained agent

(a) Define environment: In the ball-balancing environment, the RL agent interacts with a simulation where the goal is to balance the ball on the plate using the Kinova Gen3 robotic arm while avoiding obstacles. The environment is defined as follows:

Observation space: The observation space is a continuous space that is represented by a vector with 22 elements. A distinct observation is associated with each element of the vector. The elements that make up the observation vector are as follows:

1. Joint angles of the two actuated joints (continuous).
2. The two actuated joints' joint velocities (continuous).
3. Positions of the balls (x and y offsets from the middle of the plate) (continuous).
4. Ball velocities (derivatives of x and y) (continuous).
5. Plate orientations (quaternions) (continuous).
6. Plate velocities (quaternion derivatives) (continuous).
7. Joint torques since the previous time step (continuous).
8. Ball's mass and radius (constants).

Each of these components is a continuous value that reflects the status of the environment at a particular time step.

Action space: The action space is also a continuous space represented by a 2-element vector. The normalized joint torque value for each of the two actuated joints on the robotic arm is represented by an element of the vector. The action vector is continuous and ranges from -1 to 1 for each joint, where -1 represents maximum negative torque, 0 represents no torque (i.e., neutral position), and 1 represents maximum positive torque. The agent selects the continuous joint torque values as actions at each time step to control the robotic arm and balance the ball on the plate effectively.

Reward function: The reward function used in the environment comprises three components:

1. rball: A reward for the ball going toward the plate's center (continuous).
2. rplate: A penalty for plate orientation (continuous).
3. raction: A penalty for control effort (continuous).

Each of these components is a continuous value, providing feedback to the agent about its performance and behavior during the simulation. By specifying continuous observation and action spaces, the environment allows the RL agent to learn precise control strategies to balance the ball on the plate effectively. The agent can fine-tune its actions based on the continuous observations and the provided reward signal to optimize its behavior and achieve the task's objective. The number of observations and actions from the environment are numObs = 22; and numAct = 2 respectively. The continuous observation and action specifications are supported by the Simulink environment interface. A reset function is responsible for randomly initializing the ball's initial x and y positions relative to the center of the plate. This function ensures that each time a new episode starts, the ball is placed at a different position on the plate, introducing variability in the initial conditions for the simulation. It is necessary to designate Sample Time (T_s) and Simulation Time (T_f) as T_s = 0.01 s; and T_f = 10 s respectively.

(b) Create agent: The agent is a SAC agent in this simulation. To approximate the value of the policy, SAC agents use one or two parametrized Q-value function approximators. The Q-value function critic takes the present observation and action as inputs and produces a single scalar that represents the estimated discounted cumulative long-term reward that the agent would earn by implementing the strategy in the state corresponding to the present observation. The SAC agent in this example employs two critics for state and observations. The two critics have considered updating each state and action space for joint movement. The neural network used to represent each critic's parametrized Q-value function has two input layers: one for the observation channel (provided by 'obsInfo') and the other for the action channel (given by 'actInfo'). The scalar value corresponding to the Q-value estimation is returned by the output layer.

For each critic's neural network, model utilize an array of layer objects to specify each network path. Each path's input and output layers are given names as well. These names connect the pathways and expressly relate the network input and output levels to the pertinent environmental channels. By structuring the neural networks in this effective way to model the Q-value functions for the SAC agent with two critics. This design facilitates the agent's learning and decision-making processes throughout the simulation. The critic neural network is shown in Figure 6.

To implement the SAC agent with two critics, it is essential to create and initialize two dlnetwork objects. These dlnetwork objects represent the neural networks used as the critics in the SAC architecture.

SAC agents are created for continuous action spaces, and they use a continuous Gaussian actor to carry out a parametrized stochastic policy. This actor neural network creates random behaviors by selecting samples from a Gaussian probability distribution. The Gaussian actor has two output layers, each with the same number of elements as the action space dimensions. Each action dimension's mean values are provided by one output layer, while the other output layer provides each dimension's standard deviations. A Softplus or rectified linear unit (ReLU) layer ensures the standard deviations are non-negative. Importantly, the 'UpperLimit' and 'LowerLimit' parameters of 'actInfo,' utilized to generate the actor, are read by the SAC agent to determine the action range. Internally, the agent scales the distribution and restricts the action to the predetermined bounds. As a result, it is unnecessary and undesirable to include a "tanhLayer" as the final nonlinear layer in the mean output path. To view the actor neural network, one can use the plot function plot(actorNetwork). Figure 7 shows the actor network.

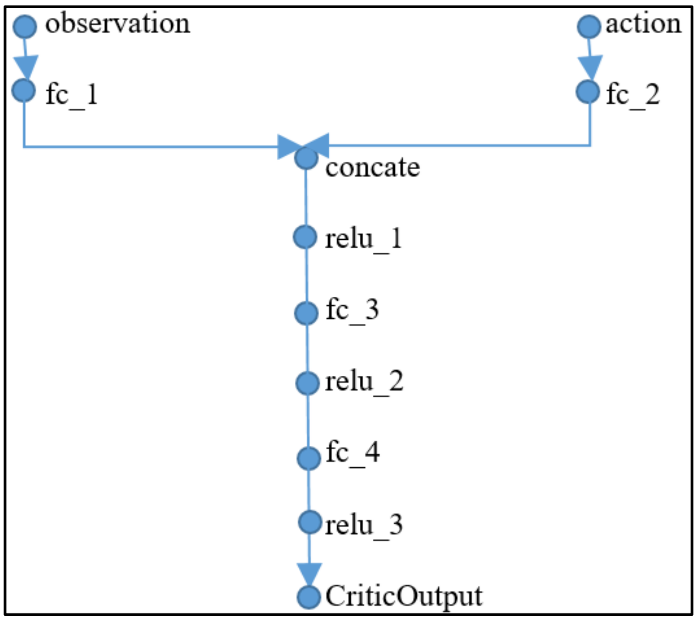


Figure 6: Critic Network

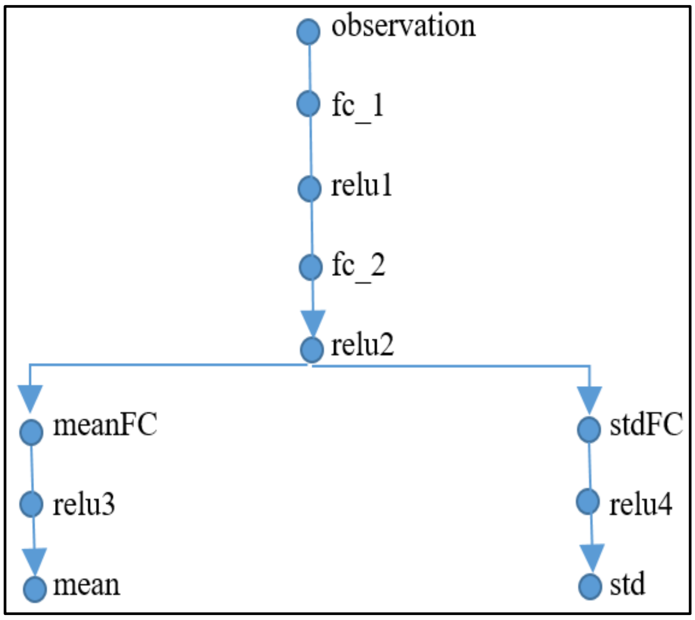


Figure 7: Actor network

The actor function is created using the rlContinuousGaussianActor class, which is designed for handling continuous action spaces with Gaussian policies. This function defines the neural network architecture for the actor, which generates actions based on observations. An experience buffer with a maximum capacity of 1 million samples is used in this simulation to train the SAC agent. Mini-batches of size 128 are randomly chosen from this buffer during training to update the agent's neural network. The agent employs a discount factor of 0.99, or about 1. A higher discount factor tends to favor long-term rewards over short-term rewards. This means the agent is more inclined to prioritize actions that lead to higher cumulative rewards over extended periods rather than focusing solely on immediate rewards. The Adam optimization technique updates the actor and critic neural networks throughout the simulation. The optimizer is set up with a gradient threshold of 1 and a learning rate of 0.0001 (1e-4). This setup ensures that the optimization process efficiently updates the network weights based on the observed gradients while preventing large gradient magnitudes that could lead to unstable training.

(c) Training an agent: For training an agent, the rlTrainingOptions is used to specify the training parameters:

1. 200 maximum episodes, each having a maximum runtime of floor (T_f/T_s) time steps.
2. Training should be stopped when the agent obtains an average cumulative reward of more than 450 over 100 consecutive episodes.
3. To expedite training, use the Parallel Computing ToolboxTM software and set the UseParallel option to true.
4. To use the Use Parallel option, make sure the Parallel Computing ToolboxTM software is installed and correctly configured. This will expedite training and increase its effectiveness.

By changing the 'doTraining' argument to false and loading a pre-trained agent to speed up the execution of this example. When the ball is released on the plate, the Kinova Gen3 robotic arm fails to balance it before training, as shown in Figure 8, and the ball falls down from the plate.

Set 'doTraining' to true if you want to start over and train the agent from scratch but be aware that this will require a lot of calculation and could take a while. The trained agent is used to complete the remaining jobs effectively for the time being.

(d) Simulate trained agent: The ball's initial position with respect to the plate's center must be provided. Set the userSpecifiedConditions flag to false to randomize the starting ball position during the simulation. For simulation configuration, create a simulation options object. For the simulation episode, the agent will only be simulated for a maximum of floor (T_f/T_s) steps.

Define initial conditions for the ball:

ball.x0 = -0.125 + 0.25 * rand; % Initial x distance from plate center (meters)

ball.y0 = -0.125 + 0.25 * rand; % Initial y distance from plate center (meters)

Figure 9 shows the status of Kinova Gen3 robotic arm when RL training starts. The Kinova Gen3 arm, when trained with SAC, is now able to balance the ball on the plate after avoiding collision with the static obstacle. To visualize the trajectory of the ball, Figure 10 can be used. This figure will display the movement of the ball over time, allowing you to observe its path and position changes.

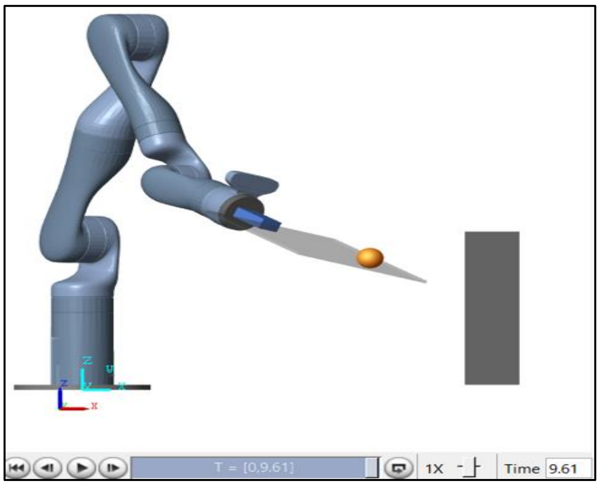


Figure 8: Kinova Gen3 robotic arm before training training starts

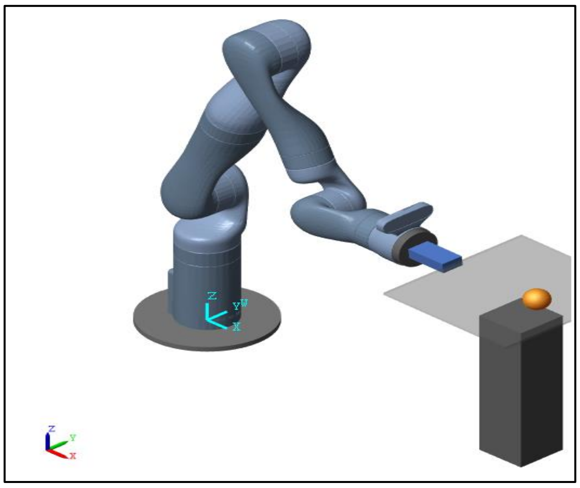


Figure 9: Kinova Gen3 robotic arm when RL

Figure 11 shows the Kinova Gen3 7-DOF manipulator attached with a plate to its end effector, and a Ping Pong ball is placed over to be balanced by the arm while manipulating through the defined RL environment. The robot is trained with a SAC algorithm to avoid collision with the grey-colored solid obstacle placed in the RL environment. After successful training, the Kinova Gen3 robotic arm is able to balance the ball on the plate without colliding with the obstacle present in the environment.

Figure 12 shows the collision avoidance of the manipulator with the black color solid obstacle present in the environment. After collision avoidance, the manipulator is still balancing the ball on the plate in the environment.

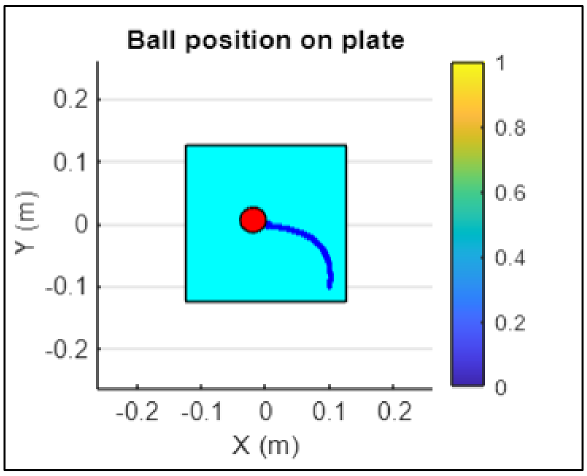
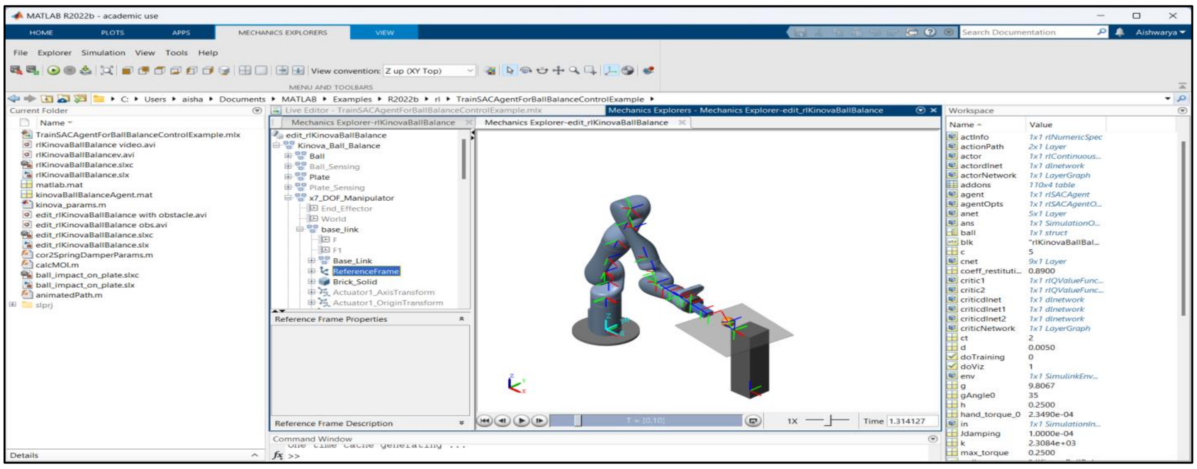


Figure 10: Ball position on plate after successful training


MATLAB R2022b - academic use
HOME
PLOTS
APPS
MECHANICS EXPLORERS
VIEW
Search Documentation
Aishwarya
File
Explorer
Simulation
View
Tools
Help
MENU AND TOOLBARS
Current Folder
Live Editor - TrainSACAgentForBallBalanceControlExample.mlx
Mechanics Explorers - Mechanics Explorer-edit_rlKinovaBallBalance
Mechanics Explorer-rlKinovaBallBalance
Mechanics Explorer-edit_rlKinovaBallBalance
Reference Frame Properties
Reference Frame Description
Command Window
Details
Workspace
1X
Time

View convention: Z up (XY Top)

C: ▸ Users ▸ aisha ▸ Documents ▸ MATLAB ▸ Examples ▸ R2022b ▸ rl ▸ TrainSACAgentForBallBalanceControlExample ▸

Name
- TrainSACAgentForBallBalanceControlExample.mlx
- rlKinovaBallBalance video.avi
- rlKinovaBallBalancev.avi
- rlKinovaBallBalance.slxc
- rlKinovaBallBalance.slx
- matlab.mat
- kinovaBallBalanceAgent.mat
- kinova_params.m
- edit_rlKinovaBallBalance with obstacle.avi
- edit_rlKinovaBallBalance obs.avi
- edit_rlKinovaBallBalance.slxc
- edit_rlKinovaBallBalance.slx
- cor2SpringDamperParams.m
- calcMOI.m
- ball_impact_on_plate.slxc
- ball_impact_on_plate.slx
- animatedPath.m
- slprj

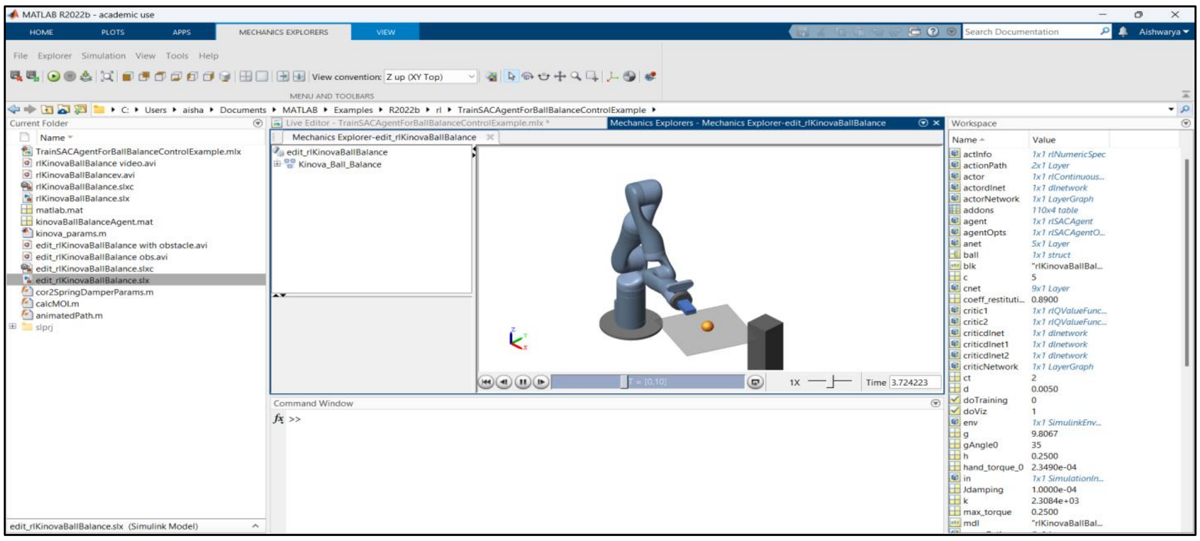
```
edit_rlKinovaBallBalance
  Kinova_Ball_Balance
    Ball
    Ball_Sensing
    Plate
    Plate_Sensing
    x7_DOF_Manipulator
      End_Effector
      World
      base_link
        F
        F1
        Base_Link
        ReferenceFrame
        Brick_Solid
        Actuator1_AxisTransform
        Actuator1_OriginTransform
```

```
One time cache generating ...
>>
```

T = [0,10] Time 1.314127

Name	Value
actInfo	1x1 rlNumericSpec
actionPath	2x1 Layer
actor	1x1 rlContinuous...
actordlnet	1x1 dlnetwork
actorNetwork	1x1 LayerGraph
addons	110x4 table
agent	1x1 rlSACAgent
agentOpts	1x1 rlSACAgentO...
anet	5x1 Layer
ans	1x1 SimulationO...
ball	1x1 struct
blk	"rlKinovaBallBal...
c	5
cnet	9x1 Layer
coeff_restituti...	0.8900
critic1	1x1 rlQValueFunc...
critic2	1x1 rlQValueFunc...
criticdlnet	1x1 dlnetwork
criticdlnet1	1x1 dlnetwork
criticdlnet2	1x1 dlnetwork
criticNetwork	1x1 LayerGraph
ct	2
d	0.0050
doTraining	0
doViz	1
env	1x1 SimulinkEnv...
g	9.8067
gAngle0	35
h	0.2500
hand_torque_0	2.3490e-04
in	1x1 SimulationIn...
Jdamping	1.0000e-04
k	2.3084e+03
max_torque	0.2500

Figure 11: Arm balancing the ball on plate


MATLAB R2022b - academic use
HOME
PLOTS
APPS
MECHANICS EXPLORERS
VIEW
Search Documentation
Aishwarya
File
Explorer
Simulation
View
Tools
Help
MENU AND TOOLBARS
Current Folder
Live Editor - TrainSACAgentForBallBalanceControlExample.mlx *
Mechanics Explorers - Mechanics Explorer-edit_rlKinovaBallBalance
Mechanics Explorer-edit_rlKinovaBallBalance
Command Window
Workspace
1X
Time

View convention: Z up (XY Top)

C: ▸ Users ▸ aisha ▸ Documents ▸ MATLAB ▸ Examples ▸ R2022b ▸ rl ▸ TrainSACAgentForBallBalanceControlExample ▸

Name
- TrainSACAgentForBallBalanceControlExample.mlx
- rlKinovaBallBalance video.avi
- rlKinovaBallBalancev.avi
- rlKinovaBallBalance.slxc
- rlKinovaBallBalance.slx
- matlab.mat
- kinovaBallBalanceAgent.mat
- kinova_params.m
- edit_rlKinovaBallBalance with obstacle.avi
- edit_rlKinovaBallBalance obs.avi
- edit_rlKinovaBallBalance.slxc
- edit_rlKinovaBallBalance.slx
- cor2SpringDamperParams.m
- calcMOI.m
- animatedPath.m
- slprj

edit_rlKinovaBallBalance.slx (Simulink Model)

```
edit_rlKinovaBallBalance
  Kinova_Ball_Balance
```

```
>>
```

T = [0,10] Time 3.724223

Name	Value
actInfo	1x1 rlNumericSpec
actionPath	2x1 Layer
actor	1x1 rlContinuous...
actordlnet	1x1 dlnetwork
actorNetwork	1x1 LayerGraph
addons	110x4 table
agent	1x1 rlSACAgent
agentOpts	1x1 rlSACAgentO...
anet	5x1 Layer
ball	1x1 struct
blk	"rlKinovaBallBal...
c	5
cnet	9x1 Layer
coeff_restituti...	0.8900
critic1	1x1 rlQValueFunc...
critic2	1x1 rlQValueFunc...
criticdlnet	1x1 dlnetwork
criticdlnet1	1x1 dlnetwork
criticdlnet2	1x1 dlnetwork
criticNetwork	1x1 LayerGraph
ct	2
d	0.0050
doTraining	0
doViz	1
env	1x1 SimulinkEnv...
g	9.8067
gAngle0	35
h	0.2500
hand_torque_0	2.3490e-04
in	1x1 SimulationIn...
Jdamping	1.0000e-04
k	2.3084e+03
max_torque	0.2500
mdl	"rlKinovaBallBal...

Figure 12: Arm balancing the ball on plate without colliding with the obstacle

Results

In recent years, there has been a remarkable surge in the application of artificial intelligence (AI) techniques within the field of robotics, particularly for complex manipulation tasks. This paper provides a comprehensive approach of SAC algorithm of RL methodology pertaining to AI-driven manipulation tasks on Kinova Gen3 Robot using MATLAB/Simulink.

The accompanying graphs, specifically shown in Figure 13 and Figure 14, offer a visual representation of the learning process of a Kinova Gen3 robotic arm over multiple episodes of a RL task. In the initial stages, it is observed that the episode rewards exhibit a decreasing trend as the arm interacts with the environment. However, as the number of episodes progresses, a notable transformation occurs. The Kinova Gen3 robotic arm begins to learn and adapt from its interactions with the environment, resulting in a substantial increase in episode rewards.

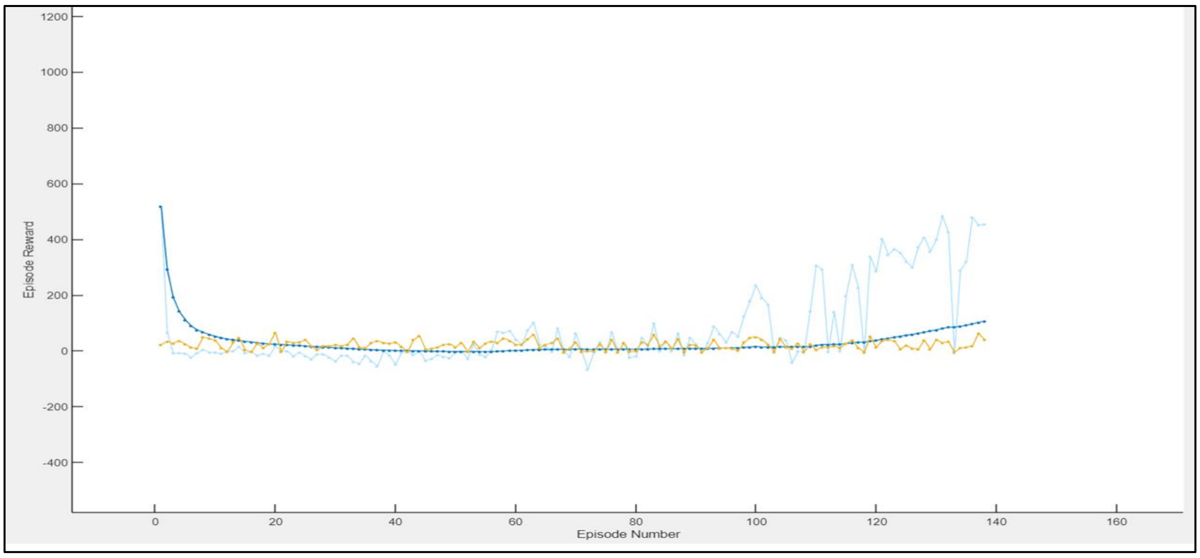
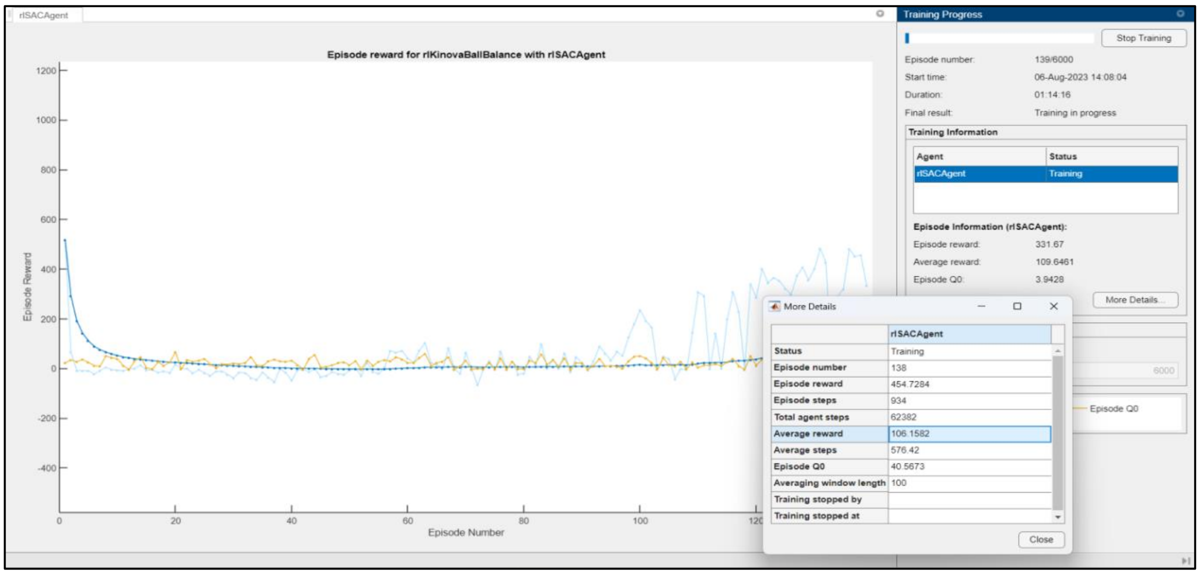


Figure 13: Graph of episode number vs episode reward



Training Progress

Episode number: 139/6000

Start time: 06-Aug-2023 14:08:04

Duration: 01:14:16

Final result: Training in progress

Training Information

Agent	Status
rlSACAgent	Training

Episode Information (rlSACAgent):

Episode reward: 331.67

Average reward: 109.6461

Episode Q0: 3.9428

More Details

	rlSACAgent
Status	Training
Episode number	138
Episode reward	454.7284
Episode steps	934
Total agent steps	62382
Average reward	106.1582
Average steps	576.42
Episode Q0	40.5673
Averaging window length	100
Training stopped by	
Training stopped at	

Figure 14: Graph of RL episode manager available in MATLAB/Simulink

Table 2. Theories and data to support the benefits of current work

Reference No.	Focus	Strength	Limitations
(5)	Internet of Skills framework, testing RL methods without demonstrations	- Improvement in performance on inverse kinematics and obstacle avoidance tasks - PPO recommended for stability - Foundation for developing an Internet of Skills	Policy proximal optimisation (PPO) recommendation, limited comparison to recent RL advancements
(28)	Indoor inspection tasks in confined areas	- Stable autonomous navigation - Safe travel configuration - Hardwired data connection	Limited information on specific tasks, future studies needed
(29)	Optimal path determination using 3D cubic Bézier curves	- Modification of path without changing control points - Hierarchical clustering for optimal path based on curvature, torsion, and path length - Works in 3D environments with different obstacles	Lack of real-world testing, focus on specific shape parameters
(30)	Torque reduction in confined spaces	- Reduction in required actuator torque with bracing contacts - Realistic deployment potential in confined spaces for collaborative manufacturing	Planning time constraints, fidelity of dynamics model
(31)	Safe robot motion in the presence of humans	- Theoretical stability properties - Productivity assessment - Potential for industrial applications - Replanning in the presence of unexpected human reactions	Lack of real-world human interaction testing
(32)	Path and joint configuration planning for redundant manipulators	- Effective solution for 3D obstacle avoidance - Combining sampling-based and optimization-based algorithms - Suitable for path and joint configuration planning	Limited exploration of optimal end-effector paths
(33)	Payload increase for a manipulator	- Use of B-spline parametric curves for smooth joint trajectories - Reliable payload increase for Kinova Gen3 - Exploration of optimization parameters for efficiency	Limited exploration of dynamic constraints, assumptions in trajectory generation

Robot simulation

The successful training process provides an evidence that the Kinova Gen3 robotic arm has achieved a profound understanding of the RL environment. This is highlighted by the arm consistently receiving maximum rewards, with some episodes even reaching a remarkable score of up to 454. The primary objective of the task is to maintain balance while delicately maneuvering a ball on a plate, all the while avoiding collisions with obstacles strategically placed within the environment. In the depicted graph, which shows the relationship between episode number and episode rewards, there is an initial decline in episode rewards. However, as the number of episodes progresses, there is a clear trend of the Kinova Gen3 robotic arm learning from its interactions with the environment, ultimately leading to the attainment of maximum rewards. In Figure 13 and Figure 14 the light blue color represents the highest episode rewards achieved in each episode, while the yellow color corresponds to the growing number of episodes conducted. The dark blue shade, on the other hand, signifies the seamless training process of the robotic arm utilizing the SAC algorithm.

Discussion

This paper only focuses on the static obstacle placed in the robot's path but when trained in different RL environments with dynamic obstacles the results may vary depending on the training episodes and parameters. The training of SAC will be further dependent on the RL environment created and differentiated according to the robot and particular task it is trained with.

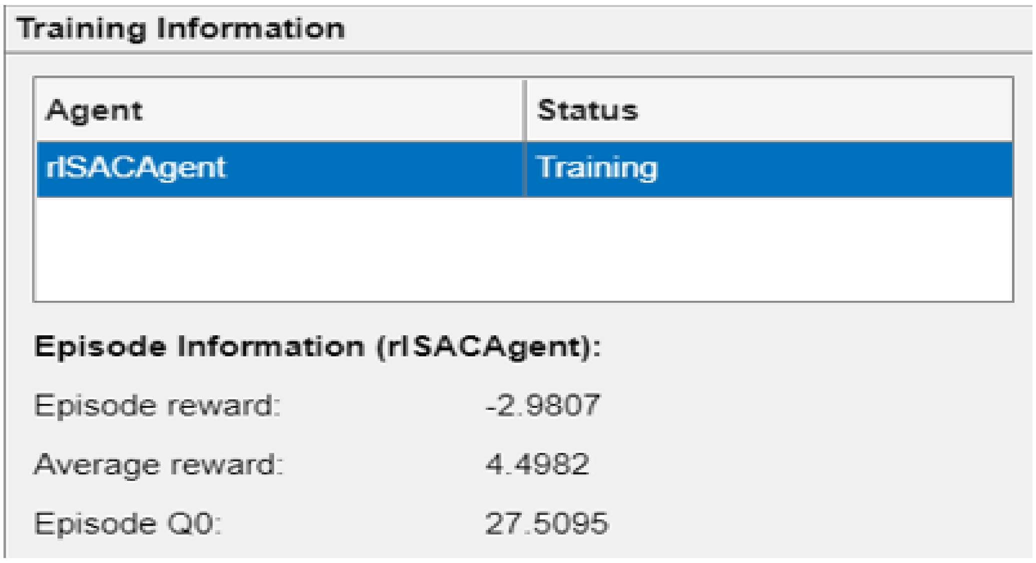
RL offers a suite of benefits for obstacle avoidance in various domains. It enables the development of robust and flexible obstacle avoidance systems that can evolve and improve their strategies over time, making them well-suited for real-world applications where environmental conditions may change unpredictably. Table 2 details about the theories and data from the previous related research work to support the benefits of current work. The research findings mentioned in the Table 2 reveal the important limitations of the related work that provide opportunities for further development in the field of obstacle avoidance for robotic arm manipulation. Some studies lack real-world trials, which are important in ensuring the effectiveness and reliability of practical algorithms and systems. Additionally, humans tend to focus on specific aspects, such as planning the right path or optimization, without having to delve into many robot tasks. Computational efficiency, especially planning time, has been cited as a limiting factor in some projects. The latest research prioritizes reducing planning time for time allocation. While some research work focuses on the use of sensors. Recent developments in this field often explore advanced technology, ML, and cognitive algorithms to increase the power and flexibility of robotic systems.

This previous related research includes different tasks and algorithms but did not focus on the obstacle avoidance in RL environments. To overcome this research gap, the RL based obstacle avoidance of Kinova Gen3 robotic arm is performed. The methodology used in this research work is based on recent developments and can reduce time required to train the robot with reduced training episodes.

In the initial training phase, the Kinova Gen3 robotic arm was not able to balance the ball on the plate and failed to avoid collision with the obstacle. The Kinova Gen3 robotic arm initially set to 6000 episodes for training but as training episodes reach to 200 the agent/robot learned to avoid collision with the static obstacle present in the environment. The maximum rewards assigned to these episodes is 454 and robots were able to achieve it in 10 seconds after completion of training.

In Figure 15 it is displayed that the robot was learning form the environment and due to that the rewards are negative. But after the successful training of the robot, as shown in Figure 16, the maximum reward of 454 were assigned to the robot which indicates that the training turned out to be successful and robot avoided the collision with the obstacle.

Term "average reward" in Figure 15 and Figure 16 typically refers to the expected sum of rewards obtained by the agent over time, averaged across multiple episodes. The average reward in SAC is a key metric used to evaluate the performance of the learned policy. It reflects how well the agent is doing in the environment over a period of time. The term "episode Q0" in Figure 14, Figure 15 and Figure 16 generally refers to the initial estimate or initialization of the Q-function at the beginning of each episode. In SAC, Q0 represents the initial estimate for the Q-function at the start of an episode before any learning updates occur.


Training Information

Agent	Status
rlSACAgent	Training

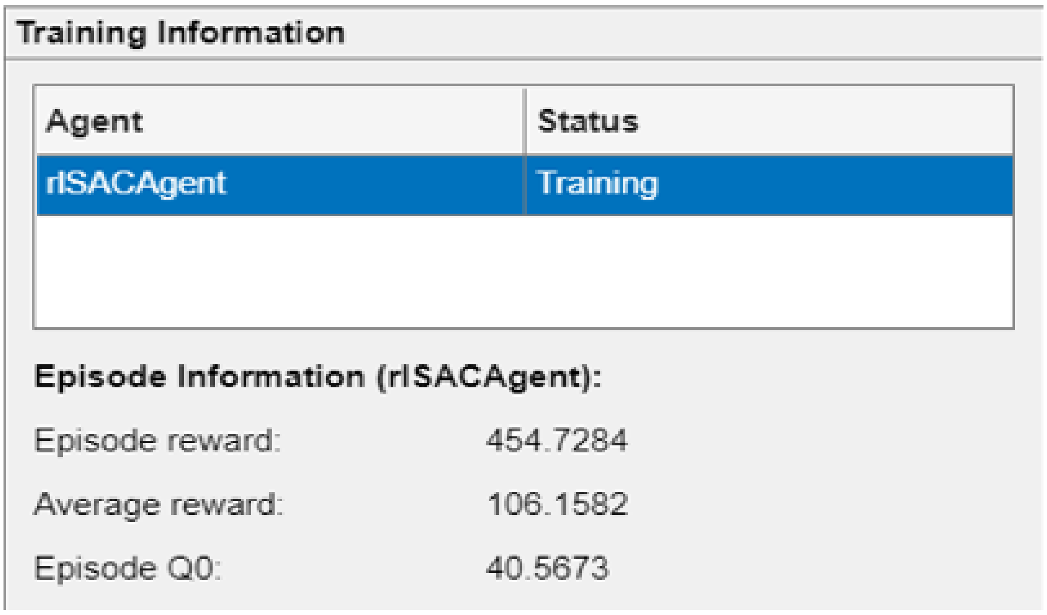
Episode Information (rlSACAgent):

Episode reward: -2.9807

Average reward: 4.4982

Episode Q0: 27.5095

Figure 15: Before SAC training


Training Information

Agent	Status
rlSACAgent	Training

Episode Information (rlSACAgent):

Episode reward: 454.7284

Average reward: 106.1582

Episode Q0: 40.5673

Figure 16: After SAC training

The Kinova Gen3 7-DOF manipulator successfully avoided the collision with the static obstacle present in the environment using the SAC algorithm based on RL. The Simulation took 10 seconds after successful training and proved to be the stable algorithm for the ball-balancing task with a maximum reward of 454. After achieving maximum rewards, the Kinova Gen3 robot manipulator performed the ball balancing task without colliding with the static obstacle present in the RL environment.

Future research may include dynamic obstacles present in the environment. This problem can be further solved with other RL-based algorithms, such as deep deterministic policy gradient (DDPG) and asynchronous advantage actor critic (A3C) algorithm, and one can compare the results.

Abbreviation

Abbreviations used in this manuscript are defined at their first occurrence in the manuscript. However, for quick reference, a consolidated list is provided as below:

7-DOF: 7-degree-of-freedom

A2C: Advanced Actor Critic
A3C: Asynchronous Advantage Actor Critic
ACER: Actor Critic with Experience Replay
ACLA: Actor-Critic Learning Automation
AI: Artificial Intelligence
API: Application Programming Interface
C51: Categorical DQN
DDPG: Deep Deterministic Policy Gradient
DQN: Deep Q-Network
DR-DQN: Delayed-Reward Deep Q-Network
HER: Handsight Experience Replay
I2A: Imagination-Augmented Agents
MBMF: Model-Based Priors for Model-Free
MBVE: Model Based Value Expansion
MCTS: Monte-Carlo Tree Search
ML: Machine Learning
NAF: Normalized Advantage Function
PG: Policy Gradient
PPO: Policy Proximal Optimisation
QR-DQN: Quantile-Regression Deep Q-Network
QV-Learning: Quality-Value Learning
ReLU: Rectified Linear Unit
RL: Reinforcement Learning
ROS: Robot Operating System
SAC: Soft Actor-Critic
SARSA: State-Action-Reward-State-Action
TD: Temporal Difference
TD3: Twin-Delayed Deep Deterministic Policy Gradient
TRPO: Trust Region Policy Optimization

Acknowledgement
Nil

Author contributions
All authors contributed equally. All authors critically reviewed and approved the final manuscript.

Conflict of interest
The authors have no conflict of interest to declare.

Ethics approval
This research work did not require ethics approval because it did not collect new data from humans, animals, or their experiments.

Funding
Nil

References

1. Javaid M, Haleem A, Singh RP, Suman R. Substantial capabilities of robotics in enhancing industry 4.0 implementation. Cognitive Robotics. 2021;1:58-75. https://doi.org/10.1016/j.cogr.2021.06.001.
2. Soori M, Arezoo B, Dastres R. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. Cognitive Robotics. 2023;3:54-70. https://doi.org/10.1016/j.cogr.2023.04.001.
3. Arafat MY, Alam MM, Moh S. Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges. Drones. 2023;7(2):89. http://dx.doi.org/10.3390/drones7020089.
4. Rodriguez-Guerra D, Sorrosal G, Cabanes I, Calleja C. Human-robot interaction review: Challenges and solutions for modern industrial environments. IEEE Access. 2021;9:108557-78. https://doi.org/10.1109/ACCESS.2021.3099287.
5. Guzman L, Morellas V, Papanikolopoulos N. Robotic embodiment of human-like motor skills via reinforcement learning. IEEE Robot Autom Lett. 2022;7(2):3711-7. https://doi.org/10.1109/LRA.2022.3147453.
6. Zhang T, Mo H. Reinforcement learning for robot research: A comprehensive review and open issues. Int J Adv Robot Syst. 2021;18(3): 172988142110073. https://doi.org/10.1177/17298814211007305.
7. Akalin N, Loutfi A. Reinforcement learning approaches in social robotics. Sensors. 2021;21(4):1292. https://doi.org/10.3390/s21041292.
8. Naeem M, Rizvi STH, Coronato A. A gentle introduction to reinforcement learning and its application in different fields. IEEE Access. 2020;8:209320-44. https://doi.org/10.1109/ACCESS.2020.3038605.
9. Shakya AK, Pillai G, Chakrabarty S. Reinforcement learning algorithms: A brief survey. Expert Syst Appl. 2023;231(120495):120495. https://doi.org/10.1016/j.eswa.2023.120495.
10. Ibarz J, Tan J, Finn C, Kalakrishnan M, Pastor P, Levine S. How to train your robot with deep reinforcement learning: lessons we have learned. Int J Rob Res. 2021;40(4-5):698-721. https://doi.org/10.1177/0278364920987859.
11. Lobbezoo A, Qian Y, Kwon H-J. Reinforcement learning for pick and place operations in robotics: A survey. Robotics. 2021;10(3):105. https://doi.org/10.3390/robotics10030105.
12. Lindner T, Milecki A. Reinforcement learning-based algorithm to avoid obstacles by the anthropomorphic robotic arm. Appl Sci. 2022;12(13):6629. https://doi.org/10.3390/app12136629.
13. Han D, Mulyana B, Stankovic V, Cheng S. A survey on deep reinforcement learning algorithms for robotic manipulation. Sensors. 2023;23(7):3762. https://doi.org/10.3390/s23073762.
14. Abdulsaheb JA, Kadhim DJ. Classical and heuristic approaches for mobile robot path planning: A survey. Robotics. 2023;12(4):93. https://doi.org/10.3390/robotics12040093.
15. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor: results of the 35th International Conference on

Machine Learning (ICML 2018), 10-15 July 2018, Stockholm. Sweden: PMLR 80; 2018. p. 1861-1870. https://doi.org/10.48550/arXiv.1801.01290.
16. D'Oro P, Schwarzer M, Nikishin E, Bacon P-L, Bellemare MG, Courville A. Sample-efficient reinforcement learning by breaking the replay ratio barrier: results of the Eleventh International Conference on Learning Representations (ICLR 2023), 1-5 May 2023, Kigali. Rwanda: openreview.net; 2023. Available at https://openreview.net/forum?id=OpC-9aBBVJe.
17. Elguea-Aguinaco I, Serrano-Muñoz A, Chrysostomou D, Inziarte-Hidalgo I, Bøgh S, Arana-Arexolaleiba N. A review on reinforcement learning for contact-rich robotic manipulation tasks. Robot Comput Integr Manuf. 2023;81(102517):102517. https://doi.org/10.1016/j.rcim.2022.102517.
18. James S, Wada K, Laidlow T, Davison AJ. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretization: results of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022), 18-24 June 2022, New Orleans, LA. USA: IEEE; 2022. p. 13729-38. https://doi.org/10.1109/CVPR52688.2022.01337.
19. Taylor AT, Berrueta TA, Murphey TD. Active learning in robotics: A review of control principles. Mechatronics. 2021;77(102576):102576. https://doi.org/10.1016/j.mechatronics.2021.102576.
20. Hutsebaut-Buysse M, Mets K, Latré S. Hierarchical reinforcement learning: A survey and open research challenges. Mache Learn Knowl Extr. 2022;4(1):172-221. https://doi.org/10.3390/make4010009.
21. Dawood M, Dengler N, De Heuvel J, Bennewitz M. Handling sparse rewards in reinforcement learning using model predictive control: results of the 2023 IEEE International Conference on Robotics and Automation (ICRA 2023), 29 May-2 June 2023, London. United Kingdom: IEEE; 2023. p. 879-85. https://doi.org/10.1109/ICRA48891.2023.10161492.
22. Zhao T, Wang M, Zhao Q, Zheng X, Gao H. A path-planning method based on improved soft actor-critic algorithm for mobile robots. Biomimetics. 2023;8(6):481. https://doi.org/10.3390/biomimetics8060481.
23. Ezzeddine F, Ayoub O, Andreoletti D, Giordano S. SAC-FACT: Soft actor-critic reinforcement learning for counterfactual explanations. In: Longo L, editor. Explainable Artificial Intelligence. xAI 2023. Communications in Computer and Information Science. Cham: Springer Nature Switzerland; 2023. p. 195–216. https://doi.org/10.1007/978-3-031-44064-9_12.
24. Banerjee C, Chen Z, Noman N. Improved soft actor-critic: Mixing prioritized off-policy samples with on-policy experiences. IEEE Trans. Neural Netw Learn Syst. 2022;1-9. https://doi.org/10.1109/TNNLS.2022.3174051.
25. Ding F, Ma G, Chen Z, Gao J, Li P. Averaged soft actor-critic for deep reinforcement learning. Complexity. 2021; 2021(6658724):1-16. https://doi.org/10.1155/2021/6658724.
26. Kinova Robotics. KINOVA Gen3 ultra lightweight robot guide. Kinovarobotics.com. 2022. Available at https://www.kinovarobotics.com/uploads/User-Guide-Gen3-R07.pdf.
27. Veit C. Kinova robotic arm manipulation with python programming. Master of Science Thesis, Florida Atlantic University, Florida; 2022. Available at https://fau.digital.flvc.org/islandora/object/fau%3A96156/datastream/OBJ/view/KINOVA_ROBOTIC_ARM_MANIPULATION_WITH_PYTHON_PROGRAMMING.pdf.
28. Pearson E, Szenher P, Huang C, Englot B. Mobile manipulation platform for autonomous indoor inspections in low-clearance areas: result of the ASME 2023 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 20-23 August 2023, Boston, Massachusetts. USA: ASME; 2023. p. DETC2023-111245. https://doi.org/10.1115/DETC2023-111245.
29. Bulut V. The optimal path of robot end effector based on hierarchical clustering and Bézier curve with three shape parameters. Robotica. 2022;40(9):3266-89. https://doi.org/10.1017/S0263574722000182.
30. Natarajan R, Johnston GLH, Simaan N, Likhachev M, Choset H. Torque-limited manipulation planning through contact by interleaving graph search and trajectory optimization: results of the 2023 IEEE International Conference on Robotics and Automation (ICRA 2023), 29 May-2 June 2023, London. United Kingdom: IEEE; 2023. p. 8148-54. https://doi.org/10.1109/ICRA48891.2023.10161297.
31. Oleinikov A, Kusdavletov S, Shintemirov A, Rubagotti M. Safety-aware nonlinear model predictive control for physical human-robot interaction. IEEE Robot Autom Lett. 2021;6(3):5665–72. https://doi.org/10.1109/LRA.2021.3083581.
32. Sepehri A, Moghaddam AM. A motion planning algorithm for redundant manipulators using rapidly exploring randomized trees and artificial potential fields. IEEE Access. 2021;9:26059–70. https://doi.org/10.1109/ACCESS.2021.3056397
33. Doiron N, Gallant A. Increasing the payload of a 7dof cobot. In: Laribi MA, Nelson CA, Ceccarelli M, Zeghloul S, editors. New Advances in Mechanisms, Transmissions and Applications. Cham: Springer Nature Switzerland; 2023. p. 101–10. https://doi.org/10.1007/978-3-031-29815-8_11.