Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 7, July 2021: 3304-3313

Industrial Robotic Arm Design enabled by AI-based Deep Reinforcement Learning using a combination of Deep Deterministic Policy Gradient (DDPG) and Hindsight Experience Replay (HER)

K. Sai Srinivas[a], R. Raman Kumar[b], Tejesh Sudheer[c], M. Somasundaram[d], T. Suresh[e]

[a,b] IV Year Student, Department of Electronics and Communication Engineering, R.M.K. Engineering College, Kavaraipettai – 601206, India.
[c] Founder, Impossible Robotics, Chennai – 600018, India.
[d]Professor, Department of Electronics and Communication Engineering, R.M.K. Engineering College, Kavaraipettai – 601206, India.
[e]Professor and Head, Department of Electronics and Communication Engineering, R.M.K. Engineering College, Kavaraipettai – 601206, India.

Abstract

Robotic arms have huge impact in Industrial process automation and help in significantly reducing the manufacturing time and cost of production. Most of the industrial robots deployed across various industries are programmed to perform repetitive tasks like welding, painting, packaging, moving or putting objects in predefined trajectories. This paper presents an implementation of additional intelligence imparted to a robotic arm for manufacturing purposes using Reinforcement Learning technique. In current scenario, the robotic arms are using inverse kinematics to operate in their environment. In general, the desired goal to reach is represented in Cartesian co-ordinates and it is converted to Joint angles using the inverse kinematics. The robotic arm should move it's joints in certain angles and reach the final position. Solving the kinematics demands high computational power due to solving of multi-dimensional matrices as a part of calculating the joint angles. It is also passive calculation and there is no feedback system to check and correct the position during next iteration. This process is also prone to errors that arise due to matrix inversions during calculation. The Deep reinforcement learning method with a combination of DDPG and HER Algorithm's using sparse rewards, proposed and implemented in this paper, uses rewards and penalty system to enable the system to learn directly from the surrounding environment and prevent any miscalculations. This process directly generates the joint angles and positions based on the intelligence attained through several training epochs and the current state is calculated based on the Q-table parameters which are computed using two different target Neural Networks known as Actor and Critic Networks.

Keywords: artificial intelligence, industrial robotics, reinforcement learning, 3D printed robotic arm, DDPG, HER, deep reinforcement learning.

1. INTRODUCTION

As the manufacturing industries are moving towards automation, usage of robotic arms has increased in the recent times in order to meet the supply demand and reduce the human labor that significantly cuts the production costs and increases the production capacity. Preparing such a robotic arm includes path planning and kinematics which are to be solved using mathematical equations and their complexity increases as the Degrees of Freedom (DoF) increase. A typical 6 DoF robotic arm also called as serial robot is commonly used in most of the industrial applications have complex kinematic equations to be solved for the movement. Sometimes user intervention is required to rectify issues like singularities and unsafe positions. These are to be solved either

manually or avoided using path planning. Though closed loop systems are used for precise movements and calibration [1], the robots cannot operate efficiently when the environment or goal is slightly changed in real time. It requires that intelligence has to be imparted to the robot by incorporating advanced sensors like depth camera sensor [2] and artificial intelligence techniques. Using the advanced sensors, the system can be made more accurate, flexible and adopt to the operating environments in real time. Artificial intelligence techniques can be used to achieve better control and take intelligent decisions in reaching the goal [3].

To prevent this manual intervention and make the robot self-aware of the surrounding environment, a goal-based reinforcement learning approach [4] is used with inputs derived from a 3D depth camera [2] that provides the system with necessary information about the goal location and it's working envelope.

2. Robotic Path/Motion Planning

Path/Motion planning is an essential part for functioning of self-autonomous industrial robotic-arm, it accesses the overall performance of the system. While path/Motion planning is implemented certain problems are needed to be taken care. The major problems include avoiding null-points, self-collisions. The above problems are handled and taken care by using a ROS moveit package [5]. Move-it is a motion planning software that can efficiently handle robotic manipulation, motion planning, inverse kinematics and detect self-collisions [5].

Robotic Kinematics is use of kinematic equations to apply geometric equations to study different movements of multi-degree of freedom kinematic chain which defines and forms the structure of a robot. When it comes to path planning there is a requirement to solve the Forward or inverse kinematics based on the application.

2.1 Forward Kinematics

Forward Kinematics is a problem that needs to be solved when we have all the positions of the joint angles to compute target position of the end effector. This is of no-use in solving Robotic-Path-planning in real-world.

2.2 Inverse-Kinematics

Solving the problem of inverse-kinematics is a very important for efficient path-planning and robotic manipulation. It is a mathematical problem which is needed to be solved to calculate variable joint parameters in order to move the end effector to the target location.

2.3 Solving Inverse-Kinematics problem using A.I.

Solving Inverse Kinematics problem is quite complex and compute in-efficient for higher degree-of-freedom robotic-arm.[6] To solve this, we moved to use deep-reinforcement learning approach where the robotic-arm is trained in a simulated environment [7] to optimize itself to a target location and then store the respected join-angles to the database and this is repeated for different target-points such that the database have the entire optimal-map for performing the inverse kinematics [17] task and path-planning if a target position is given.

Figure 1. Developed 6 axis robotic arm.

3. Current Implementation of AI in Robotic Arms

- Andrea Franceschetti in their research in 2020 'Robotic Arm Control and Task Training through Deep Reinforcement Learning' [11], had proposed a detailed comparison of Trust Region Policy Optimization and DeepQ-Network. However, our results show that a combination of both DDPG and HER has yield better results.

- Adarsh Sehgal in their research in 2020 'Deep Reinforcement Learning Using Genetic Algorithm for Parameter Optimization' [14], had proposed genetic algorithms to find the parameters used in DDPG and HER. However, No Genetic algorithms are used to find parameters instead deep neural-networks are used to find and optimize the parameters for DDPG, and HER is used in our current research.

- Kaveh Kamali in their research in 2020 'Real-time Motion Planning for Robotic Teleoperation Using Dynamic-goal Deep Reinforcement Learning' [16], had proposed a method that maps human hand motions to the robotic arms. However, we have trained the robot in a simulated environment to impart the intelligence for path planning.

- Julian Ibarz in their research in 2021, 'How to train your robot with deep reinforcement learning: lessons we have learned' [21], had conducted a study of deep reinforcement learning methodologies in different robotic environments. This study highly helped us to fine-tune our deep reinforcement learning approach.

- Ashwani K in their research in 2020, 'Evolution Of Optimization Algorithm Operated Robotic Arms With Different DOF' [20], had conducted a study of optimization of different algorithms for better robotic path planning and other aspects of robotic arms with different DOF. This study helped in choosing the better approach toward solving the current problem using deep reinforcement learning.

- Justinas Miseikis et al in their research in 2018 'Robot Localisation and 3D Position Estimation Using a Free-Moving Camera and Cascaded Convolutional Neural Networks' [8], had used the approach of replacing sensors with deep learning approach to estimate the position of the joints by converting the 3D image to 2D image with cascaded CNN. However, it is very difficult to calculate and solve inverse kinematics from a camera. Also, it is less Accurate and it would not be aware of null points in the space.

- Alexander Reiter et al in their research in 2016 'Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators' [9], had used the approach of a Velocity Jacobian based scheme working with linear combination of null space basis vectors. However, Inverse Kinematics is generated from mathematical equation solving which is not an ideal solution as it is a compute intensive and time taking method for solving larger Degree of Freedom.

- Stephen James et al in their research in their research in 2016 '3D Simulation for Robot Arm Control with Deep Q-Learning' [4] had used the approach of deep Q networks and 3D simulations to train a 7 DOF robotic arm in a task with no prior knowledge. However, the proposed method is for discrete action space , which is not ideal for real world robotics.

- Sourya Dipta Das et al in their research in 2018 'Energy Optimized Robot Arm Path Planning using Differential Evolution in Dynamic Environment' [6], had used the approach of detecting the shortest and energy efficient path for an industrial robot using differential evolution (DE) algorithm. However, this Energy optimization technique is limited to solving the differential equation but still it is energy consuming as it has to be continuously invoked for planning the path.

4. Reinforcement Learning

Reinforcement learning is used in the system as it is intended to be completely autonomous. It has modules as below:

4.1. Q-Learning

Q-learning is defined as an off policy algorithm in reinforcement learning where the agent learns optimal policy with the help of a greedy policy. In this process it finds the best action to take when there is given current state. It learns from actions that are not present the current policy i.e. by performing random actions. A Q-leaning requires a Q-table which acts as a memory table for the agent to take actions for a given state.

$$Qt(s,a) = Qt-1(s,a)+\alpha(R(s,a)+ \gamma \, maxa' \, Q(s',a') - Qt - 1(s,a)) \qquad Eq\ 1$$

4.2. Q-table

Q-table is a 2-dimensional matrix of [state, action] which consists Q- values where columns represent all the possible actions in the environment and rows represent the state of the system. A Q-value is a measure of the overall expected reward assuming the agent is in state (s) and performs action (a). Initializing of Q-table is done by setting-up Q-values to 0 or through random initialization. The Q table is updated continuously with the best action out of all possible actions at a given time instant i.e., the action that yields highest reward and the same is updated as the state value. The updating of Q- table is governed by Bellman's equation as show below.

$$Qt(s,a)=Qt-1(s,a)+\alpha(R(s,a)+\gamma max(Q(s',a'))-Qt-1(s,a)) \qquad Eq\ 2$$

Where

Qt(s,a) = New Q-value

Qt−1(s,a) = Previously Recorded Q-value

α = Learning Rate

R(s, a) = A reward function which takes a state (s) and action (a) and outputs a reward value (r)

γ = Discount Factor

max(Q(s′,a′)) - Max Q-value of the current state from the Q-table

This equation is iterated over for each episode and Q-table is updated for maximizing the reward of the agent.

This traditional Q-table approach fails to work in real time complex scenarios as the computational space increases largely and tends to reach an infinite [state action] space. So, for many of the real-world applications from robotics to self-driving cars uses deep learning-based approach to overcome this problem.

4.3. Deep Q-Learning Network (DQN)

Deep Q-Network uses a Deep Neural architecture which helps in approximating the Q- Values for a given state. The neural network further optimizes the best action for any given state by predicting the Q-value for each action and the prediction gets better over the period of training.

4.4. Improving Efficiency of a Deep Q Network

To increase the efficiency of a Deep Q Network (DQN) techniques like epsilon greedy, reply buffer, target network is employed which increase the efficiency and accuracy of the DQN.

4.4.1 Epsilon Greedy Algorithm:

Epsilon Algorithm [20] is an exploration factor which helps the network to define its rate of exploration over the period of training. Initially a high epsilon rate is set and is reduced using epsilon- decay factor until a it reaches the minimum epsilon value. This helps the agent in exploration and exploitation of the environment.

4.4.2 Replay Buffer:

To enhance the DQN performance a Replay Buffer[24] is introduced to collect and store the experience in a large buffer which contains all the transitions taken during the agent training. Data is trained in Mini Batches form the buffer. This methodology increases data efficiency in training the DQN agent.

4.4.3 Target Network:

This method employs using two individual neural networks namely online network and Target Network. Online network is used to interact with the environment whereas target network is used to predict the outcomes. The online network is constantly updated but the target network is updated over N iterations by copying the trained parameters form Online network. This improves the stability of the network.

5. Implementation of Solution

The solution is implemented using an open-source robotic arm connected to a cloud computer where our reinforcement learning algorithm [12] runs and generates the next position co-ordinates based on the robotic environment which is a 3D video captured by a camera on field that assists robotic arm manipulation to reach the goal [13]. The following block diagram shows detailed view of the entire system.

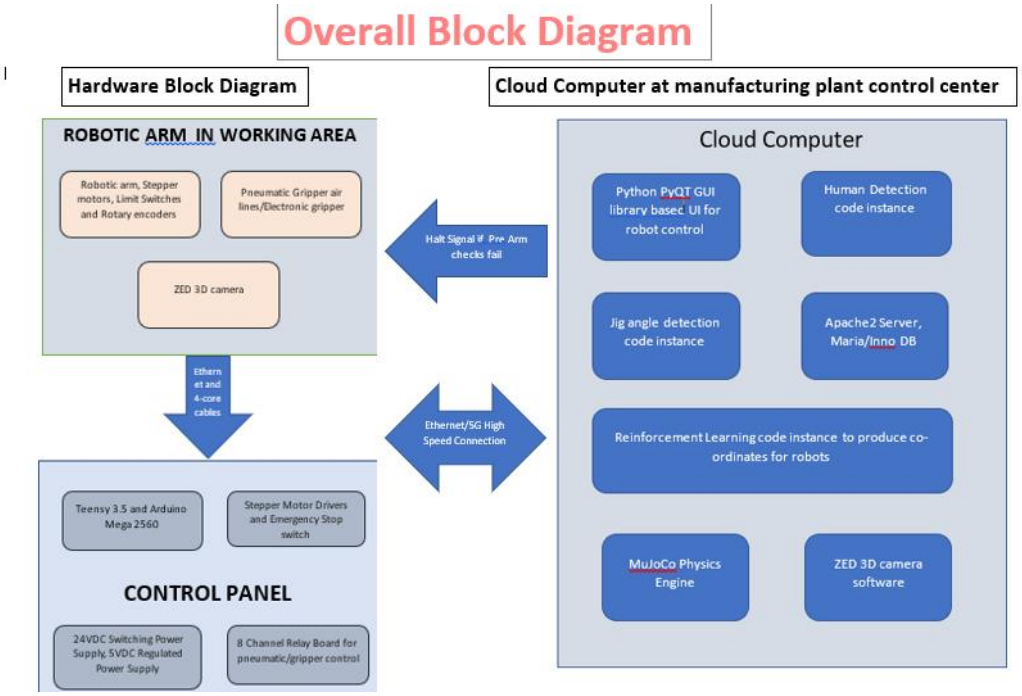

Figure 2. Detailed overall block diagram of the system

To implement the reinforcement learning algorithm in the robotic arm and use it in the real time, a simulation is carried out in the initial phase using a 3D physics engine and then the results are analyzed for accuracy and desired output.

5.1 URDF Modelling:

The Universal Robot Description Model (URDF) is prepared using the 3D modelling software for use in the simulation environment. This URDF file holds details about all the physical and mechanical properties of the robot to be simulated like locations of joints, their friction, gravity, mass, location of the part in a 3D space represented as quaternions in a XML file format [7]. The visuals of the robot are shown in the simulation software using the Stereo Lithography (STL) files that are generated using 3D modelling software like AutoCAD.

5.2 MuJoCo Simulation:

Simulation of the required process is carried out using the physics simulation [25] engine by importing the URDF file and then passing the location values and co-ordinates of the different joints to the software which then renders the simulation using OpenGL hardware accelerated graphics. This allows the user to visualize the movements of the robot and the various actions performed to reach the goal during the training phase of reinforcement learning.

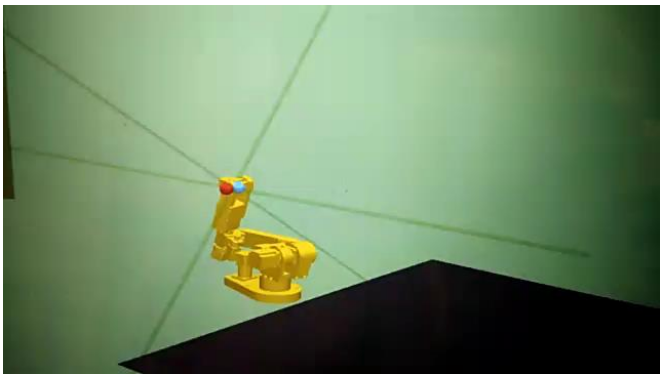

Figure 3. MuJoCo simulation of the serial manipulator

5.3 Replacement of Inverse Kinematics by the Deep Reinforcement Learning:

In current scenario, the robotic arms are using inverse kinematics to operate in their environment. In general, the desired goal to reach is represented in Cartesian co-ordinates and it is converted to Joint angles using the inverse kinematics which is further used in motion planning of the arm. The robotic arm should move it's joints in certain angles and reach the final position. The Joint space co-ordinates are calculated from the Cartesian co-ordinates using Denavit-Hartenberg parameters (DH Parameters) which are used in assigning the co-ordinate frames for the robotic arm. Solving the kinematics demands high computational power due to solving of multi-dimensional matrices as a part of calculating the joint angles. It is also passive calculation and there is no feedback system to check and correct the position during next iteration. This process is also prone to errors that arise due to matrix inversions during calculation.

The reinforcement learning method uses rewards and penalty system to enable the system to learn directly from the surrounding environment and prevent any miscalculations. This process directly explores [15] and generates the joint angles and positions based on the intelligence attained through several training epochs and the current state is calculated based on the Q-table parameters which therefore helps in real time motion planning [16]. This process imparts intelligence to the robot and prevents any problems that arise due to miscalculations in conventional method. Reinforcement learning method is superior to conventional method in terms of adopting itself to new environments based on the intelligence attained thorough training [11].

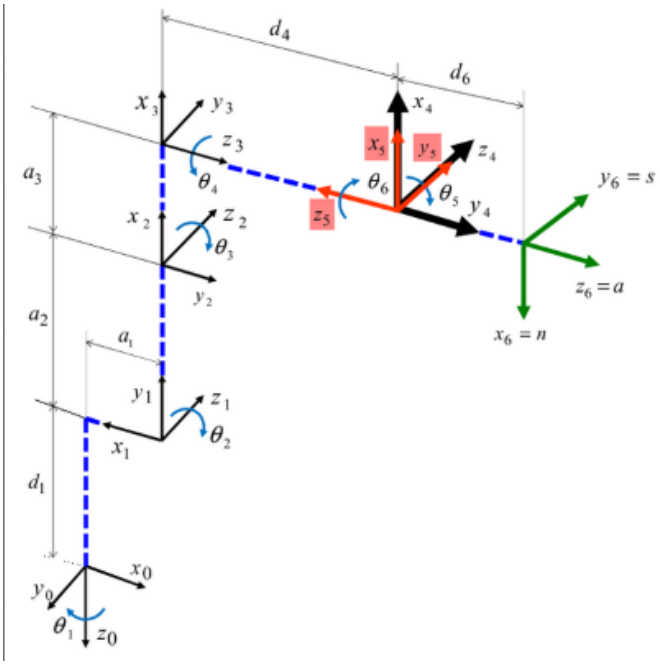

Figure 4. Kinematic chain diagram of the proposed serial manipulator.

$$\theta_4 = \mathbf{atan2}(r_{13}, r_{23})$$
$$\theta_6 = \mathbf{atan2}(-r_{31}, r_{32})$$
$$\theta_5 = \mathbf{atan2}\left(r_{33}, \pm\sqrt{1-r_{33}^{\,2}}\right)$$
$$\theta_4 = \mathbf{atan2}(-r_{13}, -r_{23})$$
$$\theta_6 = \mathbf{atan2}(r_{31}, -r_{32})$$

Figure 5. Conversion of Cartesian Co-ordinates to Joint Co-ordinates in conventional inverse kinematics

5.4 Sensors and 3D Camera for Position Detection:

The most important component in reinforcement learning is being able to see or sense it's surroundings in order to generate rewards and decide future action. This purpose is fulfilled by using a 3D camera [5] that can map the surroundings before every action and fed as input to the trained model which allows the trained model to take decisions. The 3D camera gives 3D location of the object in terms of color gradients which is processed to get the actual co-ordinates.

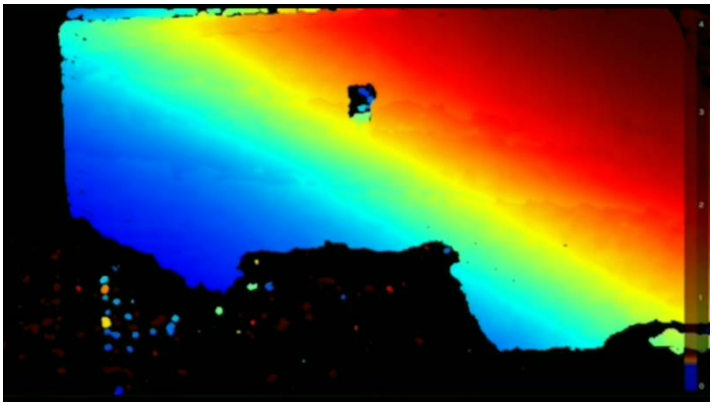

Figure 6. Depth camera view of an object depicting distances as color gradients.

Figure 7. Normal camera view of the same object from previous picture.

6. Algorithms Used:

6.1 Deep Deterministic Policy Gradient (DDPG):

DDPG[14] is a model free approach and can learn policies using low-dimensional observations with same hyper-parameters and neural network structure [4]. It uses an Actor-Critic Network similar to DQN target network implementation. The Actor-Critic network is updated for each step taken in the environment and learns from the mini-batches from the replay buffer. A β_θ policy is constructed by adding noise that is sampled from noise process (N) this makes sure that the environment is explored.

$$\beta_\theta(s_t) = \mu_\theta(s_t) + \mathcal{N} \qquad \textit{Eq 3}$$

These steps are repeated by updating the Actor-Critic network for each step.

6.2 Hindsight Experience Replay (HER):

Robotics environments require to deal with sparse rewards to keep the complexity of the network under control which is the biggest problem in reinforcement learning. HER [10] helps in dealing with the sparse rewards by using a concept of generating a virtual goal allowing sample-efficient learning [5]. HER is constructed by using experience of some episodes which are stored in the replay buffer for every transaction along with the current goal and subset of some virtual goals. This type of Goal-Engineering mechanism helps the agent to avoid false-positives and obtain learning signal to achieve our final goal.

A deep reinforcement learning[22] architecture using combination of DDPG[14] and HER[10] is used to achieve best results for our robotic environment.

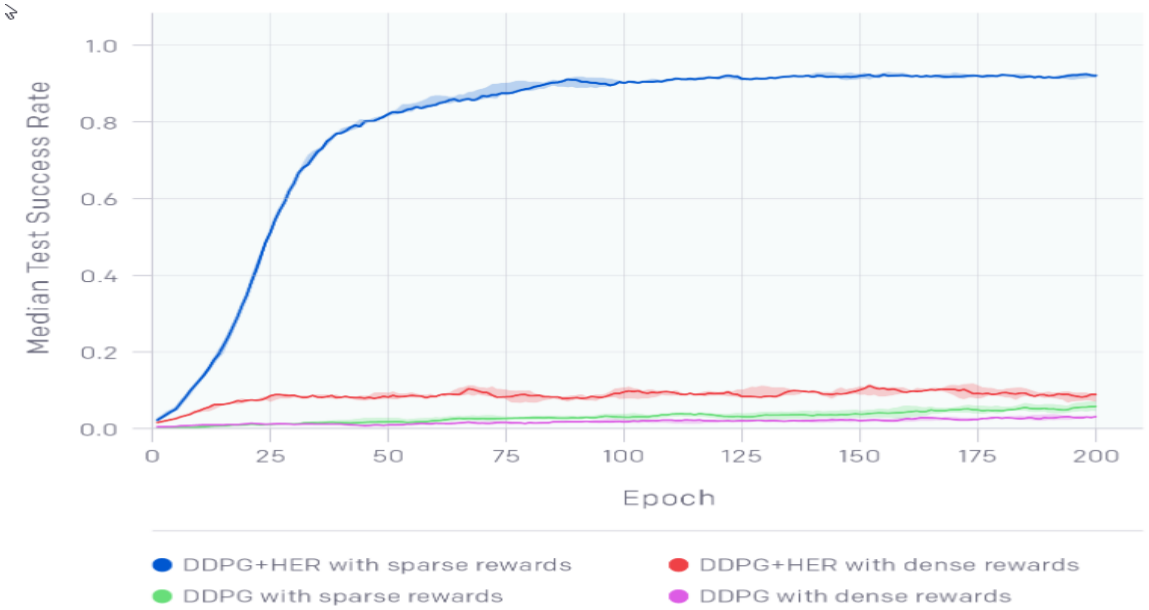

Figure 8. Graph depicting success rates of the algorithms

The above results show that a combination of DDPG+HER gives a high success rate which is tested and verified by OpenAI on the robotics environments within the OpenAI Gym [23].

The above algorithms are imported form OpenAI Baselines and modified to satisfy our own robotic environment which is built using MuJoCo(Physics Simulation) and OpenAI gym[23] python library.

7. Conclusion and Future Scope

This technology is still under research and our system has produced satisfactory results in testing phase. With the changing trends, usage of advanced technologies like reinforcement learning looks promising for industrial automation. This technology can undergo potentially major improvements and will produce better and improved versions of the existing algorithms. More and more environments can be added to the training phase to make the model work in different kinds of industrial tasks.

References

[1] Self-calibration of single-loop, closed kinematic chains formed by dual or redundant manipulators by D.J. Bennett, 1988.

[2] 3D Collision Detection for Industrial Robots and Unknown Obstacles Using Multiple Depth Images by Markus Fischer, 2009.

[3] Towards a science of integrated AI and Robotics by KannaRajan, 2017.

[4] Deep Reinforcement Learning Using Genetic Algorithm for Parameter Optimization by Adarsh Sehgal, 2019.

[5] Mobile manipulation task simulation using ROS with MoveIt by Hao Deng, 2017.

[6] A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242) by Ahmed R. J. Almusawi, 2016.

[7] 3D Simulation for Robot Arm Control with Deep Q-Learning. Stephen James, Edward Johns, 2016.

[8] A Deep Reinforcement Learning Approach for Dynamically Stable Inverse Kinematics of Humanoid Robots by Phaniteja S, 2018.

[9] Artificial Intelligence and Robotics Michael Brady, 1984.

[10] Hindsight Experience Replay With Experience Ranking by Hai Nguyen ; Hung Manh La ; Matthew Deans, 2019.

[11] Robotic Arm Control and Task Training through Deep Reinforcement Learning by Andrea Franceschetti, Elisa Tosello, Nicola Castaman, Stefano Ghidoni, 2020.

[12] Deep Reinforcement Learning for the Control of RoboticManipulation: A Focussed Mini-ReviewRongrong Liu, Florent Nageotte, Philippe Zanne, Michel de Mathelin and Birgitta Dresp-Langley, 2021.

[13] Synergy Emergence in Deep Reinforcement Learning for Full-dimensional Arm Manipulation by Jihui Han, Jiazheng Chai, Mitsuhiro Hayashibe, 2020.

[14] DDPG-Based Adaptive Robust Tracking Control for Aerial Manipulators With Decoupling Approach 2021

[15] Combining Learning from Demonstration with Learning by Exploration to Facilitate Contact-Rich Tasks by Yunlei Shi, Zhaopeng Chen, Yansong Wu, Dimitri Henkel,Sebastian Riedel, Hongxu Liu, Qian Feng, Jianwei Zhang, 2021

[16] Real-time Motion Planning for Robotic Teleoperation Using Dynamic-goal Deep Reinforcement Learning by Kaveh Kamali, Ilian A. Bonev, Christian Desrosiers, 2020

[17] A Reinforcement Learning Approach for Inverse Kinematics of Arm Robot by Zichang, Jin Huang, Wenjie Ren, Chundong Wang, 2019

[18] A study of neural network based inverse kinematics solution for a three-joint robot by Raşit Köker, Cemil Oz, Tarik Cakar, Hüseyin Ekiz, 2004

[19] Inverse kinematics learning for redundant robot manipulators with blending of support vector regression machines by Jie Chen, Henry Y K Lau, 2016

[20] Evolution Of Optimization Algorithm Operated Robotic Arms With Different DOF by Ashwani K, Vijay B, Darshan K, 2020

[21] How to train your robot with deep reinforcement learning: lessons we have learned by Julian Ibarz, 2021.

[22] Reproducible Reinforcement Learning Experiments For Robotic Reaching Tasks by Pierre Aumjaud, 2021.

[23] OpenAI Gym.arXiv1606.01540 by Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W , 2016.

[24] Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark by Fan, L , 2018.

[25] A Modular Simulation Framework and Benchmark for Robot Learning by Yu.