Deep Reinforcement Learning-based Industrial Robotic Manipulations

Muhammad Babar Imtiaz

Thesis presented for the degree of

Doctor of Philosophy

to the

Software Research Institute,

Technological University of the

Shannon

Supervisors:

Dr. Brian Lee
Dr. Yuansong Qiao

Submitted to the Technological University of the Shannon,
September 2023

Abstract

Pick and place robotic systems can be found in all major industries in order to increase throughput and efficiency. But most of the pick-and-place applications in the industry today have been designed through hard-coded, static programming approaches. These approaches completely lack the element of learning. This requires, in case of any modification in the task or environment, reprogramming from scratch is required every time. This thesis targets this particular area and introduces the learning ability in the robotic pick-and-place operation which makes the operation more efficient, and increases its strength of adaptability. We divide this thesis into three parts. In the first part, we focus on learning and carrying out pick and place operations on various objects moving on a conveyor belt in a non-visual environment i.e., without using vision sensors, using proximity sensors. The problem under consideration is formulated as a Markov Decision Process (MDP). and solved by using Reinforcement Learning (RL). We train and test both model-free off-policy and on-policy RL algorithms in this approach and perform their comparative analysis. In the second part, we develop a self-learning deep reinforcement learning-based (DRL) framework for industrial pick-and-place of regular and irregular-shaped objects tasks in a cluttered environment. We design the MDP and solve it by deploying the model-free off-policy Q-learning algorithm. We use the pixelwise-parameterization technique in the fully connected network (FCN) being used as the Q-function approximator. In the third and main part, we extend this vision-based self-supervised DRL-based framework to enable the robotic arm to learn and perform prehensile (grasping) and non-prehensile (non-grasping, sliding, pushing) manipulations together in sequential manner to improve the efficiency and throughput of the pick-and-place task. We design the MDP and solve it by using the Deep Q-networks. We consider three robotic manipulations from both prehensile

and non-prehensile category and design large network of three FCNs without creating any bottleneck situation. The pixel-wise parameterization technique is utilized for Q-function approximation. We also present the performance comparisons among various variants of the framework and very promising test results at varying clutter densities across a range of complex scenario test cases.

Acknowledgements

A doctorate of four years is not just an educational degree but a long journey filled with amazing and astonishing lifetime experiences. At this stage today, I would like to express my heartfelt gratitude to all those who have contributed in this journey leading to the successful completion of my doctoral thesis.

First and foremost, I am deeply indebted to my respected supervisor, Dr Brian Lee, for his significant expertise, patience and unwavering commitment to excellence and while shaping this research. His continuous guidance, constructive criticism, encouragements and never-ending support despite the complications like Covid-19 pandemic.

I am grateful to my co-supervisor Dr. Yuansong Qiao for being always there to provide guidance, suggestions, and working support.

Finally, bundle of thanks to my family for supporting me and fighting the time difference of 4-5 hours throughout these years and to my SRI family of fellow postgraduates for making this long journey a pleasant and wonderful experience.

List of Publications

Published Papers

Muhammad Babar Imtiaz, Yuansong Qiao, Brian Lee, "*Implementing Robotic Pick and Place with Non-Visual Sensing Using Reinforcement Learning*", 2022 6th International Conference on Robotics, Control and Automation (ICRCA), Xiamen, China, 2022, pp. 23-28, DOI: 10.1109/ICRCA55033.2022.9828993.

Muhammad Babar Imtiaz, Yuansong Qiao, Brian Lee, "*Comparison of Two Reinforcement Learning Algorithms for Robotic Pick and Place with Non-Visual Sensing*", International Journal of Mechanical Engineering and Robotics Research, Vol. 10, No. 10, pp. 526-535, October 2021. DOI: 10.18178/ijmerr.10.10.526-535.

Muhammad Babar Imtiaz, Yuansong Qiao, Brian Lee, "*Prehensile Robotic Pick-and-Place in clutter with Deep Reinforcement Learning*", 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET), Prague, Czech Republic, 2022, pp. 1-6, DOI: 10.1109/ICECET55527.2022.9873426.

Muhammad Babar Imtiaz, Yuansong Qiao, Brian Lee, "*Prehensile and Non-Prehensile Robotic Pick-and-Place of Objects in Clutter Using Deep Reinforcement Learning*", Sensors 2023, 23, 1513. DOI: 10.3390/s23031513.

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CAD	Computer Aided Design
CEM	Cross Entropy Method
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network
DOF	Degrees Of Freedom
DOPE	Deep Object Pose Estimation
DRL	Deep Reinforcement Learning
EST	Expansive Space Trees
FCN	Fully Convolutional Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
GQ-CNN	Grasp Quality Convolutional Neural Network
GUI	Graphical User Interface
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
MAE	Mean Absolute Error
MCQ-L	Modified Connectionist Q-Learning
MDP	Markov Decision Process
MSE	Mean Square Error
ODE	Open Dynamics Engine
OMPL	Open Motion Planning Library
PLC	Programmable Logic Controller

PPR-Net	Pointwise Pose Regression Network
PRM	Probabilistic Roadmap Method
QT-Opt	Q-learning Training – Off Policy Training
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RGB-D	Red Green Blue-Depth
RL	Reinforcement Learning
ROS	Robot Operating System
RRT	Rapidly-Exploring Random Trees
RRT-Connect	Rapidly-Exploring Random Trees-Connect
SARSA	State-Action-Reward-State-Action
SBL	Single-Query Bi-Directional Lazzy
SDF	Signed Distance Fields
SGD	Stochastic Gradient Descent
TD	Temporal Difference
URDF	Unified Robotics Description Format
V-REP	Virtual Robot Experimentation Platform
YOLO	You Only Look Once

Table of Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

1.1 Background

The term ‘Industry 4.0’ emerged around a decade ago and its whole world revolved around the aim of automation of the industry [1, p. 0]. It gradually introduced the industry to various leading-edge technologies such as Digital Twins, Cyber-Physical Systems, and Internet of Things (IoT), etc. The future of this industrial revolution has been very well-explained by prominent scholar Warren G. Bennis in 2016 [2] when he said that “The factory of the future will have two employees: a human and a dog. The task of the human will be to feed the dog. The dog will have the task to dissuade the human to touch the automated systems”.

A common objective of Industry 4.0 is automation through the use of artificial intelligence (AI), robotics and the latest novel sensors. With the help of automation, the efficiency of the industrial operations can be improved, the operational costs can be reduced and the product quality can be enhanced. In addition to these advantages, even the safety of the workplace environment can be made better by preventing any workplace injuries due to the automation. The robotic automation of any industrial process means utilizing robots to perform a certain task that would otherwise be performed manually by human labor. A vast range of industrial tasks, ranging from simple assembly to complex manufacturing processes, can be performed by the robots after programming them. In today’s industry, most of the robotic automation

is hard-coded, where the robot is programmed in a static manner to perform tasks such as material handling, assembly, packaging etc. without having any ability to learn the task dynamically. This lack of learning ability needs the robots to be reprogrammed every time in case of any modification in the task. Introducing learning ability in these robotic arms is a key research area among the research community nowadays in order to avoid the reprogramming overhead in case of any changes in the task or working environment.

The most common and important industrial robotic manipulation is pick-and-place. A number of robotic solutions are currently deployed in various industries for the pick-and-place task. Existing robotic manipulation approaches work with vision systems, which with the help of latest sensors, can utilize advanced features such as full-color high megapixel resolution, 3D graphical data, etc. to improve the manipulation process. **However, there are open questions and challenges.** Upon detailed literature review and analysis of the application areas, we were able to highlight following research gaps:

- **Tackling Non-visual Environments:** There are some such industrial environments where vision-based sensors are not an option due to different factors. These factors can be shortage of installation space or high rate of dust or vibrations[3]. In some scenarios, even wash-up from the water jets at the scene is one of the major reasons to avoid vision sensors[3]. In such environments, learning of the pick-and-place is not an easy task. Most of the literature discusses various approaches using learning technologies such as deep learning and deep reinforcement learning for the pick-and-place operations with the help of vision sensors only. The learning algorithms for non-visual environments where some

other types of sensors are used such as proximity sensors still need to be addressed.

- **Tackling Vision-based Environment without additional overhead:** A number of approaches have been presented for learning the pick-and-place tasks in vision-based environment. Most of these approaches involve additional overhead due to use of techniques such as segmentation and singulation. These techniques were necessary in previous approaches because those approaches considered all objects to be picked-and-placed individually turn-by-turn. Not much work has been done to avoid these additional overheads. Therefore, such algorithms need to be designed where an alternate route is taken in a memory efficient manner without incurring additional overhead.
- **Learning combination of robotic manipulations for pick-and-place in clutter:** Literature shows that there a number of robotic manipulations such as hand manipulations, grasping, pushing, stacking, target finding, dual block-stacking, etc. Grasping is commonly used to perform the pick-and-place task. But in the case of objects in clutter, grasping alone is always not the answer. The learning of sequential combination of different robotic manipulations needs to be focused in order to improve the pick-and-place the objects in clutter.

Reinforcement learning (RL) is one of the primary techniques used to enable the agent to learn in an interactive environment. As the name suggests, the key goal of an RL agent is to learn, which is achieved by discovering a policy through sequential decision-making while maximizing the expected reward. RL gained vast popularity as it was found successful in learning various complex tasks such as playing board games [4] and video games [5]. Despite their complexity, RL agents mastered these games by deploying the

deep neural networks as function approximators. This utilization of deep neural networks in RL framework is known as Deep Reinforcement Learning (DRL). RL-based approaches have been presented in the past to address the learning and performance of various robotic manipulations previously such as hand manipulations [6], [7], grasping [8], [9], pushing [10], [11], stacking [12], target finding [13], [14] and dual block-stacking [11], [15]. **This study deals with the use of reinforcement learning to improve the performance of pick-and-place robotic manipulation through addressing the open challenges identified above.**

1.2 Research Question & Contributions

The focus of this thesis is to answer the following ***research question***:

"Is it possible to improve the performance of vision-based pick and place systems through the use of reinforcement learning?"

The question is broken down into the following ***research objectives***:

- Study existing state of art pick-and-place systems
- Develop a reinforcement learning-based algorithm to learn and perform the pick-and-place task in a non-visual environment.
- Develop a deep reinforcement learning-based algorithm to learn and perform the pick-and-place task in a vision-based environment.
- Develop a deep reinforcement learning-based algorithm to learn and perform different robotic manipulations together.
- Document and publish the research findings.

The abovementioned research objectives were fulfilled through the following contributions of this research thesis.

Contribution I:

Development and comparative analysis of RL-based off-policy and on-policy temporal difference algorithms for industrial pick-and-place with non-visual sensing [16] [17].

This presented approach allows the agents to learn the pick-and-place task through reinforcement learning in such industrial environments where vision sensors are not viable due to certain factors. These factors can be shortage of installation space or high rate of dust or vibrations. In some scenarios, even wash-up from the water jets at the scene is one of the major reasons to avoid vision sensors. Most of the previously existing studies in this research area address the pick-and-place robotic manipulation through vision sensors only.

Therefore, this proposed algorithm addresses this research gap by enabling the agent to learn the pick-and-place task with the help of the ray-type proximity sensors instead of the vision sensors. We have trained and tested both off-policy (Q-learning) and on-policy (SARSA) temporal difference algorithms. The results show that Q-learning play better role in our proposed solution. Through this approach, the agent successfully learns to select the best suitable XYZ coordinates for the operation in accordance with the varying positions, orientations of objects on the conveyor belt and random conveyor belt speeds.

Contribution II:

Development of a DRL-based algorithm for industrial pick-and-place of regular and irregular shaped objects in clutter with the help of vision sensors [18].

This proposed framework enables the agent to learn and perform industrial pick-and-place of regular and irregular shaped objects in the clutter. There are various existing vision-based pick-and-place studies. But majority of them focus on first identifying the objects individually through different segmentation and singulation techniques resulting into additional overhead. Some other approaches require beforehand geometrical knowledge of the objects or domain specific knowledge. Our approach considers the whole workspace as one instead of dealing with objects individually using the pixelwise-parameterization technique and doesn't require any sort of beforehand domain specific knowledge or any geometrical data about the objects. We utilize the off-policy temporal difference Q-learning algorithm in our approach. Instead of dealing each object in the clutter individually, with the help of pixelwise-parameterization technique, success probabilities (Q-values) are generated for each and every pixel of the workspace. In the RL world, these success probabilities are known as expected future reward. This success probability of a pixel means the chances of success if the grasping action is carried out at that particular pixel location.

Contribution III:

Development of a DRL-based prehensile and non-prehensile robotic manipulation framework for industrial pick-and-place of regular and irregular objects in the clutter with the help of vision sensors[19].

This is an extended framework from our previous DRL-based pick-and-place approach. This extended approach enables the agent to learn and perform the industrial pick-and-place of regular and irregular shaped objects through sequence of prehensile and non-prehensile manipulations with the help of vision sensors. In existing literature, some approaches that tend to combine prehensile and non-

prehensile motion can be witnessed. For instance, in such a way that prehensile motion (grasping) is performed during the non-prehensile motion or using non-prehensile motion (pushing) to move the object to certain locations for already known grasping policies. The common limitations of most of these existing approaches are their need of beforehand domain specific knowledge such as pretrained policies or handcrafted information and their requirement to perceive objects individually with the help of segmentation and singulation techniques. We address these both limitations with the help of pixelwise-parameterization technique as explained before.

Our approach addresses three actions grasping (prehensile), left-slide (non-prehensile) and right-slide (non-prehensile) by training three individual end-to-end memory-efficient variants of the DenseNet-121 and ResNet-101 for pixelwise function approximation together simultaneously. Any approach in the past using these networks without proper and comprehensive memory management unlike us, suffers from bottleneck situation at the GPU end as it experiences the quadratic feature growth with time. Our approach also enables the agents to learn bi-directional non-prehensile manipulation (left and right) as compared to the unidirectional and heuristic dependent previous approaches.

1.3 Thesis Outline

Part I:

Chapter 2 presents the background knowledge of reinforcement learning and robotics in the industries. Section 2.1 presents the background of reinforcement learning; Section 2.2 discusses the vital role of robotic arms in the industry; Section 2.3 presents the background and comparative analysis of the available robotic simulators.

Chapter 3 consists of 3 sub-sections and describes the works that are highly related to the robotic grasping and pick-and-place: Section 3.1 reviews the existing works on the model-based robotic grasping schemes; Section 3.2 describes the existing works on the model-free robotic grasping schemes; Section 3.3 illustrates the prehensile and non-prehensile robotic schemes from the available literature.

Part II:

Chapter 4 (Contribution I) presents a RL-based learning framework for robotic pick-and-place in non-visual industrial environment. This chapter addresses our objective of developing RL-based off-policy and on-policy algorithms for learning and performing the non-visual pick-and-place tasks. The foundation laid in this chapter also provides the basis of the vision-based approaches (Chapter 5 and Chapter 6) presented in this thesis.

Chapter 5 (Contribution II) illustrates a DRL-based learning framework for robotic pick-and-place of regular and irregular-shaped objects with the help of vision sensors in an industrial environment. This chapter addresses our objective of developing DRL-based algorithms for learning and performing the vision-based pick-and-place tasks. The foundation laid in this chapter also provides the basis of the vision-based multiple robotic manipulation (Chapter 6) presented further in this thesis.

Chapter 6 (Contribution III) describes the a DRL-based learning framework for robotic pick-and-place of regular and irregular-shaped objects using both prehensile and non-prehensile robotics manipulations with the help of vision sensors in an industrial environment. This chapter addresses our objective of developing DRL-based algorithms for learning and performing the vision-based pick-and-place tasks by combination of different robotic manipulations.

Part III:

Chapter 7 presents the conclusions and future works.

Chapter 2

2 Technology Overview

2.1 Reinforcement Learning

The term reinforcement learning emerged in engineering literature for the first time in 1960's. In this area of machine learning, a problem is solved by taking sequential decisions through following a policy which is learnt while the process of expected reward maximization as shown in Figure 1. Initially, the RL agent observes the environment and its current state. Then agent performs one of the actions from the available list and ends up in the next state while earning the reward. The key point behind this whole practice is to make the agent able to select the optimal action every time so that the total accumulated reward is maximized.

As this life cycle of the agent interacting with its environment continues, this forms a stochastic process known as Markov Decision Process (MDP). A MDP consists of a tuple (S, A, δ, R, γ) [20]. The state space comprising of all the states in the environment is represented by S. The list of all available actions is denoted by A. When an agent performs an action and ends up in the new state it is called transition and the function that controls it is known as transition function and is denoted by δ. The rewards earned by the agent while performing actions and transitioning from one state to another are controlled by the reward function, denoted by R. The last element of the MDP tuple is known as discount factor which is represented by γ. The discount factor tells the agent to whether prefer the learning of the immediate rewards or the distant rewards in the future by discounting the

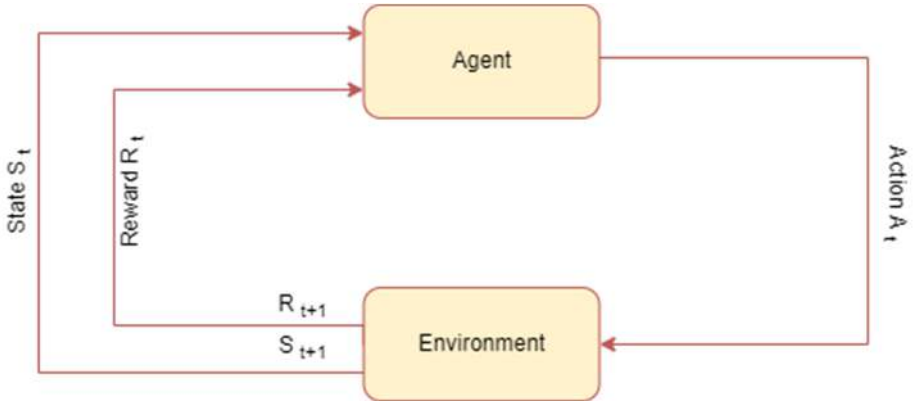

Figure 1 Reinforcement Learning Cycle

earned rewards. The value of γ is kept between 0 and 1. So that if value is kept at 0 the agent will turn completely blind to the future rewards and only prefer the immediate rewards and vice versa. In nutshell, the RL agent at given state, $s \in S$, follows the policy $\pi(s|\theta_\mu)$ and performs an action, $a \in A$, from the available action space where θ_μ represents the policy parameters. Once the action is performed, the agent transitions to the new state, $s' \in S$, with the help of the transition function, $s' = \delta(s,a) : SxA \rightarrow S$, while earning a reward r = $R(s,a) : SxA \rightarrow S$. At any given timestamp t, the main objective is to maximize the expected return, $E_{(st,\, at)\sim\pi} \sum_t \Upsilon^{t-1} R(s_t, a_t)$..

2.1.1 Temporal-Difference Learning

One of the key concepts in RL is temporal-difference (TD) learning. TD learning can be seen as amalgamation of the ideas from Monte Carlo methods and Dynamic Programming (DP). In TD learning, an agent learns from realistic experimentation with the environment instead of following a given model such as transition table. Therefore, TD learning is known as model-free learning. As it is not bound by any model, it is able to facilitate a large number of state-action pairs. The agent learns as

it experiences the environment and has no clue about the results of its actions beforehand. In TD learning, the agent is allowed to learn at every timestep as the updating process is done at every timestep instead of the episode's completion only. With each update the agent adjusts its older estimate as in Equation 1 in [20] :

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate] \quad (1)$$

where $[Target - OldEstimate]$ is known as the 'target error'. The target is calculated in this manner with the help of frequent updating.

In short, in TD learning, the agent is introduced to the environment without having any heuristic about states, actions, rewards, or any model to follow. With each interaction with its environment the agent learns, because immediately after each interaction it updates its existing knowledge. We usually divide the TD methods into two categories off-policy TD methods and on-policy TD methods.

2.1.1.1 Q-Learning

In 1989, a major breakthrough was made in the field of RL when the off-policy model-free TD algorithm Q-learning was developed by Watkins [21]. In this algorithm learning is accomplished through performing action selection according to another policy. In off-policy algorithms, there is a policy which decides how the agent will behave in the environment, what action will be chosen at any state. This policy is known as behavior policy. Meanwhile, there is another separate policy being evaluated and improved known as estimation policy or target policy. The Bellman equation represents this as follows

$$Q^{\pi}(s,a) = E[r_{t+1} + \Upsilon r_{t+2} + \Upsilon^2 r_{t+3} + \cdots | s,a] \quad (1)$$

$$Q^{\pi}(s,a) = E_{s'}[r + \Upsilon Q^{\pi}(s',a') | s,a] \quad (2)$$

Where E and Υ denotes the expected return and discount factor. In Q-learning the key objective is maximization of the Q-value with the help of policy and value iteration. In policy iteration the policy is evaluated and improved in a continuous

manner. In policy evaluation, the greedy policy achieved from the last policy improvement is used to estimate the value function V, whereas the policy is updated with actions that will increase the V to the maximum level for each state in the policy improvement part of the loop. This loop continues until the convergence point is reached while all the updates are done through the Bellman equation. With the help of the optimal Bellman equation value function *v* is updated in value iteration, which can be stated as

$$v_*(s) = \max_a E\left[R_{t+1} + \Upsilon v_*(S_{t+1}) | S_t = s, A_t = a\right] \quad (3)$$

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \Upsilon v_*(s')] \quad (4)$$

2.1.1.2 SARSA

In 1994, a new variant of existing Q-learning algorithm was introduced by Rummery and Niranjan [22]. At that time, it was name as 'Modified Connectionist Q-Learning (MCQ-L)' and later it was termed as SARSA, being abbreviation of state-action-reward-state-action. Actually, SARSA is a model-free on-policy TD algorithm. The on-policy TD algorithms such as SARSA have same policy for action selection and the policy which is being evaluated and improved at parallel. In other words, unlike Q-learning algorithm, the behavior policy and the estimation/target policy are same. At any timestep *t*, the RL agent in state *s* performs an action *a*, earns reward *r* and ends up in new state *s'* to perform a new action *a'*. The updation is done as follows

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) +, \Upsilon[R_{t+1} + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (5)$$

2.1.2 Deep Q-Learning

As we have seen in previous sections, Q-learning algorithm learns the Q-values through the process of trial-and-error. Q-values are initialized, an action is selected and performed by the RL agent, reward is received and Q values are update accordingly. It is evident that in the start randomness is a big factor but as the cycle continues and RL agents thoroughly explore and interact with the environment optimal Q-values are achieved. This whole concept is mathematically concluded in the form of the Bellman equation.

This Q-learning practice seems good in a small state space and limited actions list scenario. But what if scenario changes and a large state and action space is involved? For instance, any game where agent in current state has next possible 1000 states and 1000 actions to select from. This leads to a transition table of around million cells. This state space is still a small one considering complex games such as Chess and Go. Moreover, Q-learning cannot help with the unknown states because Q-values can't be calculated from existing states data. So how to tackle such scenarios and make Q-learning magic working here? This is where step into a new domain known as Deep Reinforcement Learning (DRL).

As the name states, DRL is a blend of the deep learning and reinforcement learning ideas. As in case of enabling Q-learning for large state space scenarios, neural networks from the deep learning domain are borrowed for the approximation of Q-value instead of a transition table consisting of millions of cells. The implementation of this idea lead to the DeepMind's Deep Q Learner algorithm, the master of Atari games[23]. So, in Deep Q-learning, the Q-value function approximation is achieved through the utilization of the neural networks. The neural network is fed the current state of the agent and in return it outputs the Q-values for all the possible available actions. The action corresponding to the biggest Q-value is chosen for execution then. Another beneficial thing to notice here is that the

application of neural network in Deep Q—learning is not restricted to fully connected networks only instead any kind of network can be utilized such as recurrent, convolutional, etc. according to the needs and requirements. The Figure 2 shows the key difference between simple Q-learning and Deep Q-learning.

2.2 Industrial Robotic Arms

An industrial robotic arm constitutes of several parts where each part has own functions and helps other parts working in synchronization. Some parts can be mobile, moving from between places with the help of wheels and some parts can have gliders allowing them to move overhead from one plane to another. But out of all, the most critical element of an industrial robot is the robotic arm. All major work is handled by the robotic arm such as pick-and-place, sorting, lifting, welding etc.

2.2.1 Parameters of Industrial Robotic Arms

Today, around eight key parameters are used to define the industrial robotic arms are as follows

- **Number of Axes**: This number tells about the flexibility of the robotic arm in the terms of movement. It is usually also represented as degree of freedom (DOF). Most of the robotic arms have 2 or more axes. Majority of robotic arms being used today are 6 DOF or 7 DOF. As the number of axes increases, more functionalities can be expected from the robotic arm.
- **Working Space**: It is the area of the environment of the robotic arm with which it can fully interact without risking any kind of collision with any object.

- **Working Envelope**: This is the whole space that the robotic occupies when it stretches to its maximum ability.
- **Payload**: It is the amount of load/weight the robotic arm can process/lift without breaking down during the process.
- **Motion Control**: Within a particularly defined area of workspace, these movements are beforehand designed to operate for instance movement through the action of sliding and rotating the joints.
- **Compliance**: This measure, upon application of force, defines the angle or distance the robotic joint will cover.
- **Repeatability**: This measure defines the capacity of the robotic arm to perform the same task over and over without compromising the accuracy, speed and efficiency of the process.
- **Drive**: This is the measure of the power generated by the motor of the robotic arm required for performing the tasks. Usually, it has a gear

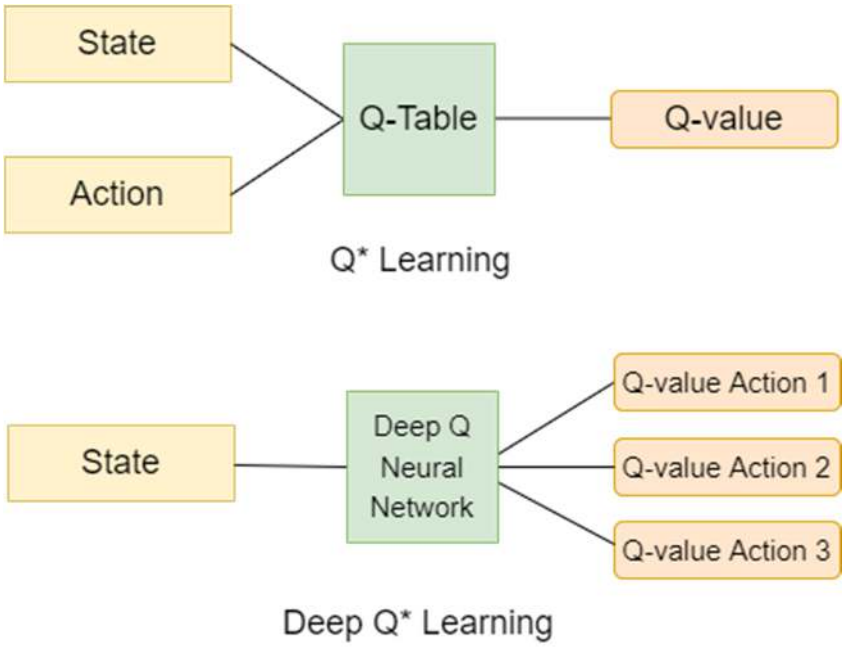

Figure 2 Q-learning vs Deep Q-learning

system in which one motion sparks the next one, constructed in a sophisticated manner leading to harmonious system.

2.2.2 Uses of Industrial Robotic Arms

In today's industry, the robotic arms deployment is increasing day by day. These industrial robotic arms are utilized for a number of functions in manufacturing such as follows

- **Material Handling:** The pick-and-place automation is a very crucial task in the process of manufacturing. The materials can be of large weight or may cause danger for humans while handling. It is also to be ensured that the whole process of pick-and-place goes completely collision-free. The pick-and-place task is a repetitive job and requires a certain speed for this repeatability without losing the accuracy. Therefore, the industrial robotic arms become the right solution for the task at hand. Robotic arms can be seen around various industries moving tons of weight around, which may otherwise have required a lot of manpower.
- **Machine Tending:** In an assembly line, a number of people are required to unload the raw materials for processing required in the assembly line to manufacture the final product. This results in huge loss of time and wastage of manpower. To tackle this issue, a machine tending robotics arm will take over and save much time and manpower without trading off the accuracy and efficiency. As the process becomes smooth enough, the production process also becomes seamless.
- **Automated Painting:** Robotic arms are utilized for painting job because human hand fails to create a uniform texture while painting and lacks the required smoothness level. For instance, a paint coat on a freshly

assembled car, requires optimal quality of action for not only aesthetic appeal but also protection against rusting problems. This optimal quality can only be ensured by a robotic arm not a human hand.

- **Assembly:** Industries such as electronics, aerospace, automotive require a lot of assembling things together. Mostly, the parts to be put together are quite heavy and prone to error if the whole process relies on humans. For such scenarios, assembly line robotic arms are designed. These robotic arms have almost negligible margin of error in operation and also have ability to manage heavy loads without compromising on the demand of speed and accuracy.
- **Automated Welding:** In industries such as aerospace and automotive, spot and arc welding are the most crucial jobs. With the help of spot and arc welding moving machineries like cars etc. are held together in one piece. This process of welding involves a high magnitude of heat and poses danger to human workers, therefore robotic arms are preferred. In manufacturing sector, these welding robotic arms make up more than 50% of all the robotic arm jobs.

2.2.3 Benefits of Industrial Robotic Arms

Within last few years, the robotic arms deployment in industries has increased by many folds. This upgradation is not by accident. It is due to the following benefits that robotic arms are now swarming in the industry

- **Safety:** As the number of the robotic arms increased and the efficiency of robotic vision maximized, the number of the accidents in the industry decreased much due to less human works involvement. This modern world industrial revolution also leads to less chances of being sued and paying

huge amounts to an injured human worker. This degree of safety of industrial plants is expected to rise more as more advanced robotics arms are coming into play.

- **Speed:** Satisfying the customer demand is not an easy task now a days. The consumption rates all around the globe are going sky, putting serious dent to the supply potentials. The industrial manufacturers require new techniques to increase the production without bargaining in the quality. The high-speed robotic arms are the right solution. Their factor of repeatability at faster rates makes them a valuable addition to the setup and at the same time also fails the human workers to compete.
- **Consistency:** In manufacturing industry, quality control is of crucial importance. Raw materials being procured day by day from different sources may vary from each other, but it is manufacturer's responsibility to even out any such variations so that the quality of finished products remain high. For such keen perfection, robotic arms are the best option. Their repeatability factor provides identical production feature leading to consistency and no fluctuations among different batches of the same product.
- **Accuracy:** When robotic arms are relied upon for manufacturing, there is almost zero room for error left as opposed to error-prone production by human workforce. These robotics arms are fully programmable and repeat the task over and over using same procedures and techniques without compromising at all on the quality. This leads to the perfect accuracy score practically and also increased customer appreciation when they get identical fully accurate products available to them.

- **Increased Production:** Once you combine the factors of safety, speed, consistency and accuracy the resultant factor you get is is increased productivity. All the delaying factors common in human workforces are successfully avoided and uninterrupted unsupervised work supply is made possible by deploying industrial robotic arms. The amount of work expected from a number of human workers in number of days is delivered within hours by the robotic arms. This high production is critical for meeting up with the high demands of market.
- **Delicate Roles:** In industries such as electronics, medical and pharmaceutical, industry cannot take chance to tarnish their reputation by producing defective items because along with their brand reputation, many lives are also at stake. For the production of extremely sensitive and delicate parts such as electricals parts, sensors or miniaturized medical materials, the industry cannot not rely on human workforce. For this level of work, only highly specialized robotic arms can be trusted.
- **Flexibility:** One of the key benefits of the robotic arms is their flexibility. Doesn't matter how specialized they are made, still one can reprogram them at any degree, even completely from scratch for a new role in the same industry. This quality kills the redundancy factor, making a robotic arm available for various roles throughout its lifespan of years and years.
- **Collaborative:** In those areas of industry, where human labor involvement is still required, they are made to work alongside the robotic arms. Such industrial robotic arms are known as collaborative robotic arms. Collaborative robotic arms have shown that they can increases the workability among the human labor and are safe to work with. This

environment of collaborative robots working among human labor also addresses the fears of humans' jobs loss due to automation of industries.

2.3 Robotic Physics Simulators

The robotics research is quite dependent on the physics simulators. These simulators are considered as a right place for testing the built theoretical models and frameworks without requiring the real physical robotics which can be rare, expensive and fragile in nature. The physics simulators instead provide cheap virtual environment where tester can design testbed of their own choice and interact with the robots. Moreover, simulations prove to be much faster than actual physical testing, can be run parallel or reset in no time unlike the real-world testing setups.

A robotic physics simulator of today can be defined as a software application designed for the end-user having following features

- Includes physics engine which aids in physical phenomenon modelling
- Includes a Graphical User Interface (GUI)
- Includes the ability to import any meshes and 3D scenes
- Includes 3D models of robots and availability of joints, actuators, sensors
- Includes collision detection mechanisms and friction modules
- Includes APIs for various programming languages such as Python/C++ etc.

Generally, the robotic simulators can be divided into different categories on the basis of their field of application. These categories include mobile robotics, marine robotics, medical robotics, aerial robotics, general manipulation and soft robotics, etc. A number of physics simulators fall under these categories designed specially to address the specific needs according to the respective field.

But here we discuss the simulators designed to help the feature of learning in the robotics. Within the last decade, learning in robotics have become a hot research area for the whole scientific community in the world. Majority of the RL algorithms suffer from the sample inefficiency problem and may have large state space and action space. While exploring such state spaces the real robots are vulnerable to execution failure or physical damage during the process. Therefore, most of the RL and DRL research works are initiated from physics simulators and then later transferred to the real robots. These learning practices can be introduced in any field which involves robotics. For instance, learning can be introduced in soft robotics. In such case, the respective simulator should not only be able to do the policy learning but also manage the soft contact ability while dealing with the materials.

There is a very famous framework for RL algorithms training and evaluation known as OpenAI Gym [24]. It usually provides it readymade environments in a simulator called MuJoCo which are generally considered for comparing performance of the new RL algorithms [25]–[27]. MuJoCo [28] is a simulator well-known in research for its ability of stable contact. The major disadvantage of MuJoCo is the absence of support for inverse kinematics and motion/path planning features. OpenAI Gym supports a number of other simulators too making them effective for the learning tasks. Gazebo [29] is one of these popular simulators being used in wide range of robotic applications including manipulations, mobile legged and wheeled robots. It provides the Robot Operating System (ROS) interface which makes the testing process easier in terms of control and also enables its transition from the simulator to the actual robot in a smooth manner. It also has an integrated library of inertial measurement unit (IMU), GPS and vision sensors. In Gazebo one can also import robotic models through Universal Robot

Description Format (URDF) files and external environments from various digital models, OpenStreetMap and SDF meshes. This simulator can run simulation for multiple robots at a time. There is a downside in a manner that it lacks the motion/path planning module but thanks to the ROS integration, ROS path planners can be utilized for the motion/path planning tasks.

Another popular simulator like Gazebo is called CoppeliaSim which was previously known as V-REP[30]. Same like Gazebo it has wide range of application such as manipulation, mobile robotics, aerial robotics, etc. It is also classified as a rigid-body simulator, means it can support multiple robots at same time. It has a large built-in library of robotic models and numerous sensors such as camera, visual sensors, stereo camera, event camera, 2d/3d laser, accelerometer, gyro, GPS, etc. The motion/path planning functionality is provided in this simulator through the OMPL library. It also provides the support for forward and inverse kinematics functionality. Built-in APIs for python/C++ languages and ROS supporting modules add a lot to the overall functionality of the simulator.

Webots is an open-source simulation application which has a wide range of sensors available in it such as cameras, GPS, LIDARS, accelerometers, IMU, radars, etc. A variety of environments, sensors and robotic models are available in this simulator. Maps importing can also be done in this simulator through the openDrive format. Included environments can be made more realistic through bringing in data from OpenStreetMap. One of these simulators is the Pybullet simulator [31]. Pybullet simulator is used in robotics research revolving around object collisions, deformable objects manipulation, pick and grasp dynamic, etc. Its key features helping the researchers include its support for forward and inverse kinematics, virtual reality functionality and RL environments. Raisim [32] is another physics simulator developed by ETH Zurich. It is not a as rich as other

simulator in terms of features but still it has played an important role in the development of high-fidelity contact dynamics models which are crucial for transition of controllers from simulation to the real-world.

In the field of robotic learning, there a number of parameters which play vital role in making simulators adequate for learning the tasks. These parameters usually include external force application on the robot, force sensors, RGBD & LiDAR sensors, availability of different physics engines, and realistic rendering abilities. A generalized comparison of these robotic simulators in terms of learning is given in Table I.

Table I Comparison of robotic simulators

Simulator Name	External Forces	Force Sensor	RGBD LiDAR	Physics Engines Variety	Realistic Rendering
MuJoCo	✓	✓	✓	×	×
Gazebo	✓	✓	✓	✓	×
CoppeliaSim	✓	✓	✓	✓	×
Webots	✓	✓	✓	×	×
Pybullet	✓	✓	✓	×	×
Raisim	✓	×	✓	×	✓

—

2.4 Summary

RL algorithms are being used in the engineering domain since early 1960's. An agent is trained through a rewarding scheme in order to learn the optimal policy for optimal action selection. In RL domain, a sub-domain is known as TD learning. In TD learning, an agent is made to learn from realistic experimentation

with the environment instead of following a given model such as transition table. Therefore, TD learning is known as model-free learning. As it is not bound by any model, it is able to facilitate a large number of state-action pairs. TD algorithms can be classified into on-policy (SARSA) and off-policy (Q-learning) algorithms. For complex problems with large state spaces, Deep Q-learning is used where a neural network is utilized for Q-values approximation.

Industrial robotic arms are increasing day by day now. They are being used to perform a number of functionalities such as material handling, machine tending, automated painting, assembly, automated welding, etc. These are the complex problems that we can make our robotic arms learn through the different learning technologies such as RL and DRL. For such learning practices, a number of physics simulators are now available such as Gazebo, CoppeliaSim PyBullets, MuJoCo, Raisim, Webots etc. These simulators have numerous sensors available in them along with multiple physics engines. Many other features such as motion/path planning, forward/inverse kinematics, collision detection/avoidance etc. are loaded in most of these simulators.

In this thesis, we tend to make our RL and DRL agents learn manipulation tasks i.e. pick-and-place of industrial robotic arms by training and evaluating them in physics simulator.

Chapter 3

3 Literature Review

The deployment of robotic arms in industry has led to a number of benefits such as safety, speed, consistency, accuracy, increased production, flexibility, human-robot collaboration, etc. Many different robotic pick-and-place approaches have been proposed so far. This chapter aims to present a comprehensive overview of the various robotic pick-and-place approaches in the existing literature.

These robotic grasping approaches can be classified according to different logical criteria. A **first criterion** , generalization, classifies robotic pick-and-place approaches as either analytic/geometric [33] or data-driven/empirical methods [34]. The analytic or geometric approaches are those in which first the geometrical shape of the object is analyzed in order to identify suitable grasping poses. Whereas, the data-driven or empirical approaches are the ones dependent on machine learning algorithms. These data-driven approaches have gained much hype in last few years. These data-driven approaches have shown more progress as there have been advances in the algorithms and computational resources along with the recent rise in data availability. In this chapter, we also focus more on the data-driven/empirical approaches instead of the old analytical methods.

A **second categorization** of robotic pick-and-place approaches is as model-based and model-free. The model-based and model-free pick-and-place approaches can be differentiated on the basis of utilization of previously known information about the objects. If the approach already possesses and exploits some specific information about the objects under consideration such as a CAD model or some

other model in order to gain the required results, it is known as model-based approach. On the contrary, if no specific information about the objects is utilized for gaining the required results, it is known as a model-free approach. Usually the model-based approaches first perform the pose estimation process for grasping and then place the object. Whereas, in the model-free approaches direct grasp candidates are proposed for the grasping and then generalization can be achieved for the novel objects.

A **third criterion** that can be adopted for categorization of the model-driven robotic pick-and-place approaches can be the type of machine learning an approach involves such as reinforcement learning, supervised deep learning, etc. The data can be manually annotated by humans or self-annotation can be achieved where automatic label generation is done. Such approaches can be further classified into two classes. In first type, known as discriminative approaches. the grasping candidates are sampled initially and then fed to a neural network which outputs the ranking list [35], [36]. Whereas, in the later one, known as generative approaches, direct generation of eligible and suitable grasping poses is done [37], [38]. The approaches can also be differentiated on the basis of their training environment i.e., simulation, real-world or both. Some approaches may work in open-loop manner and some in closed-loop manner [39]–[41]. If feedback is taken and utilized to perform the correction of the trajectory of the robotic arm it is known as open-loop approach, and if no feedback is involved it is known as closed-loop approach. If continuous visual feedback is utilized it is known as visual servoing [41]. The approaches can also be differentiated on the basis of hardware such as gripper type i.e., suction, two-finger or three finger gripper or degree of freedom such as 4DOF or 6DOF. Some approaches can target pick-and-place of individually lying single object whereas some may consider multiple objects packed in a clutter. In the same

manner, some approaches can be about directly grasping the object for the pick-and-place but some approaches can also perform some pre-grasping manipulations in order to get better chance at the grasping.

3.1 Model-based Robotic Grasping

Usually a model-based robotic grasping approach has three phases. In the first and second phase object pose and grasp pose estimation are done respectively. In third phase a collision-free kinematic path planning is done to grasp the object [42], [43]. This first part of estimating the rotation and translation of multiple objects in the workspace relative to frame of reference such as the camera is very crucial to the whole task. There are number of different factors which make this part difficult such as variations in light conditions, noise in sensors data, clutters and obstructions, and novel objects in the real world. Moreover, another issue to address is the ambiguities in the poses due to different symmetries of the objects [44]–[47]. These varying symmetries lead to different annotations even for same identical observations.

Until the system performance reaches such a satisfactory level where it becomes adaptable to the novel objects, the approaches require object-specific knowledge in large amount [48]. The required object-specific knowledge includes the template parameters, and to achieve the pose estimation the feature matching method is also required [49], [50]. The defining and tuning of grasp poses in the real-world scenarios is also required [43]. Therefore, the model-based approaches target to reduce the manual input and get automatic configurations without requiring any tunings to be made by experts so that transition to novel objects can be achieved rapidly.

In autonomous robotic manipulation, 6D pose estimation is the process in which the object's 6D pose stating its location and orientation is required. In order to benefit from the supervised learning technique in 6D pose estimation large amount of labelled data is mandatory for the training phase. The process of gathering large dataset of 6D poses of objects and then annotating them is a very time consuming and difficult process along with having the scalability issues [51]. This issue is usually addressed by working with simulations as there is plenty of synthetic data availability in simulations and the annotations are also automatically available as the ground truth. Later on various transfer techniques are available to perform the transition from simulations to the real-world [52], [53].

In last few years, most of the approaches for 6D pose estimation have been based on the utilization of the convolutional neural networks (CNN). Some of these approaches resolve the 6D pose estimation task using regression [52], [54], [55] whereas other approaches perform the discretization and classification of the pose space respectively [56], [57]. An approach, known as deep object pose estimation (DOPE) [53], first takes a RGB image as input, create 3D bounding boxes of the objects, acquire 2D image coordinates of those objects and then deploy a perspective-n-point (PnP) algorithm [58] for 6D pose estimation for each single object. This whole DOPE model is completely trained in simulations on synthetic data and later its transition from simulated world to real-world is achieved through photorealistic rendering technique along with the domain randomization [59].

For the fair comparison and transparent evaluation of state-of-the-art pose estimation approaches the benchmarking systems [60] and relevant challenges [61], [62] play a very vital role. The task of pose estimation of multiple objects in a clutter is a challenging problem and requires much focus. This scenario is usually experienced in industrial pick-and-place from bin scenarios as the clutter forms

obstruction in the pose estimation task. Such a bin-picking challenge was organized during IROS 2019 [62] where a very huge dataset of labelled 6D object poses from both simulated and real-world was used for training purposes [51]. For the evaluation of the approaches to the challenge, metrics presented by Brégier et al. [45], [47] were deployed which consider all objects having visibility more than 50% and is accounted for all the object symmetries.

Most of the learning-based approaches are able to learn the plausible object pose configurations, therefore prove to be potentially successful to the obstructions and occlusions due to clutters. The challenge mentioned above was won by an approach named PPR-Net [54]. The PPR-Net approach works at the point cloud level. It deploys a feature learning technique known as PointNet++ [63] for the pose estimation of each point in the point cloud and later finalize the pose estimation after performing the clustering and then averaging each cluster. An approach better than PPR-Net in terms of precision was presented in [52] where the pose estimation task was taken as regression task on the noisy dataset from [45]. This approach proved to be much faster as compared to the PPR-Net because it doesn't need any kind of post processing and provides very firm parameterization. This approach performs the discretization and regression of the 3D space and pose respectively. One of the major advantages of learning-based pose estimation techniques is that no manual parameter tuning is needed for the novel objects [49], [50]. Another advantage is that they can be purely trained only on synthetic data through physics engine simulators, for instance placing objects in random positions and orientations in a bin for bin-picking [51] or placing objects anywhere in scene for normal pick-and-place [53].

3.2 Model-free Robotic Grasping

The model-free approaches for robotic grasping have become very popular in recent years. The reason these model-free approaches have become the center of research community's attention is their power of generalization which allows them to entertain the unseen objects. The key factor behind growth of this particular area in robotic grasping domain is the zero requirement of object-specific knowledge. As there is no object-specific knowledge involved, there is no pose estimation phase involved which is completely opposite to the mode-based robotic grasping techniques. The model-free approaches mostly successfully adapt to the unknown novel objects and are trained in an end-to-end manner. Therefore, these approaches are well-suited to the pick-and-place tasks.

3.2.1 Supervised Learning Approaches

These are those robotic grasping learning-based approaches in which non-linear learning is achieved through the labelled/annotated training data. In this category of robotic learning, we can further classify approaches into classes known as discriminative or generative approaches. The differentiation between the discriminative approaches and generative approaches is based upon the fact that grasp configuration is being considered as input of the algorithm or being generated as the output of the algorithm.

3.2.1.1 Discriminative Approaches

In the discriminative approaches, initially sampling is performed on the grasping candidates, for instance using the cross entropy method (CEM) [64] and then they are ranked in order with the help of a neural network. The final grasping candidate

selected for the execution of robotic grasp is the one that scores the highest. These approaches usually don't play very well in terms of duration, as for identifying the high-quality grasp candidates multiple forward passes of the neural networks are to be executed on the runtime. However, these approaches still prove their worth by evaluating a number of grasps because there are no limitations due to not being restricted to the discretization of grasp space only. Moreover, for the betterment of the grasping success rate, a refinement method can also be defined on the basis of gradient [65].

Another learning-based approach was introduced in 2016 by Levine et al. [66] which focused on the hand-eye coordination in the robotic grasping. In this study, within a time period of two months, 14 robotic arms were deployed to gather labels for around 800,000 grasping candidates. First a grasping candidate is given to the trained CNN and then the current state of the bin in the form of a RGB image, and then the CNN as results predicts the success rate of the particular grasping candidate in that particular state of bin. And in the next stage it helps to servo the gripper accordingly for a successful grasp. This approach proves its worth by showing its learning capability but also lacks in a way that if there are any hardware modifications, the CNN requires a new dataset to be gathered for the retraining purpose. As stated above, it took two months to gather the dataset for training of the CNN, therefore this shortcoming can lead to a repetitive hectic and tedious process.

An approach known as Dex-Net [35], [67] was proposed in which objects place on a plane in random poses were grasped in a simulated environment. The success label of the grasping candidate along with a cropped depth image of the location of the grasping is entered as an entry in the dataset. The CNN in this approach known as Grasp Quality Convolutional Neural Network (GQ-CNN) is trained on this dataset. After being trained, grasping candidate and the depth images are provided

to the GQ-CNN which outputs the success rate of the grasping candidate as result. Moreover, it also generalizes to new rigid or flexible objects which were unknown during the training phase. Authors have extended the same Dex-Net frameworks to dual-arm robot [68] and the robotic arm with suction grippers [36] too. In extension to dual arm robot, one arm has suction gripper and the other one has parallel jaw gripper. Now the algorithm learns to use which arm in order to successfully pick the whole clutter in the bin. Moreover, another study [69] presents a fully convolutional network (FCN) system in order to generate successful grasp candidates in a cost-effective manner so that the costly procedures of sampling and ranking the grasping candidates can be avoided.

3.2.1.2 Generative Approaches

The generative approaches are those approaches where grasp candidate is the output of the algorithm. In this approach, one technique is known as robotic grasp detection. In this technique the potential grasping candidates for parallel jaw grippers are proposed through the detection of rectangles [70] in the RGB image. In this rectangle detection all parameters such as width of open gripper, location and orientation of the object are considered. However, a downside to this approach is that it is simply copy of object detection techniques [71]–[73] with an additional parameter of gripper orientation.

In order to address the scenario of the grasping of an individual object in a plane, a system was proposed by Redmon et al. [38] in 2015. This system was named SingleGrasp system. This system was capable of taking a RGB-D image as an input and then generating oriented rectangle for grasping candidate and also performing the classification of the object as the output with the utilization of a neural network.

Afterwards, another system named MultiGrasp was also proposed which used to generate more than one poses for grasping of the object as in the real-world it is possible to grasp an object in various ways. This idea of multi-grasping is the one which laid the basis of You Only Look Once (YOLO) object detection technique [71], [73]. A learning-based study [74] presented by Lenz et al. in 2015 comprises of two-stage technique where in the first stage sampling of grasping candidates is performed and in the second stage these grasping candidates are ranked with the help of another neural network. The study shows that this approach can be utilized for the real-world scenarios. The performance and success rate of such approaches can be enhanced and increased through the deployment of more sophisticated and efficient neural networks [39].

An open-source dataset known as Cornell grasping dataset [75] is available for the robotic grasping. This dataset covers around 280 objects. It has around 1035 images with manually annotated human grasps. But as the dataset is small in size, therefore heavy expansion is required to get satisfactory results [38]. On the other hand, there is also a large dataset for robotic grasping known as Jacquard dataset [76] which contains more than 50,000 images of more than 11,000 different objects and their grasping poses have been gathered from simulations. Due to large volume of samples, accurate grasping poses collected from simulation and large variety of objects the degree generalization to unseen and novel objects is high.

With the help open-source public datasets, Generative Grasping Convolutional Neural Network (GG-CNN) approach [37], [77] has been proposed which not only generate the grasping candidate but also the estimation for all the pixels of the RGB image with the help of fully convolutional network (FCN) architecture. As the FCN designed is small in size and demand low computation resources, this approach is very suitable for closed-loop grasping in non-static or dynamic scenarios. Moreover,

this proposed approach has the training part done on individual isolated images but still it is capable to work in clutter scenarios because the operation of convolution is completely local.

There is an approach [78] proposed by Zeng et al. in 2019 in which the robotic arm learns to throw various objects to given locations in order to enhances the reachability factor of the robotic arm. This approach is named as TossingBot where an end-to-end system is proposed through which the parameters of grasping and throwing are learnt together from images of the bin in trial-and-error fashion. In this manner, the system gradually learns in self-supervision mode the grasping poses which result in effective throwing. For the throwing part, simplification is achieve through only predicting the value of release velocity. A physics engine is used for the estimation of the release velocity and further adjustment tuning is done through the neural network.

The generative approaches are considered faster than the discriminative approaches because they only require a single forward pass of the neural network unlike the discriminative approaches. Moreover, they mostly provide multiple grasping candidates and the one with the optimal quality is then finalized by the robotic arm for the execution of robotic grasp.

3.2.2 RL-based Approaches

The deep reinforcement learning (DRL) branch has recently produced promising techniques for learning policies in the trial-and-error fashion. Through these DRL-based techniques, the raw data from the variety of sensors such as RGB images, depth images etc. can lead to learning and performing complex behaviors.

In clutter-based scenarios, rearranging the objects in order to enhance the grasping chances, shifting or pushing as pre-grasp robotic manipulations [79], [80] is very

important. RL-based approaches [79], [80] have been use to learn policies which not only learn the pre-grasp manipulations leading to better grasping opportunities but also exhibit the generalization property to deal with novel objects. But these approaches attempt to utilize complex FCNs but lack in proper memory management thus leading to bottleneck situations.

A comparison study of RL-based robotic grasping techniques has been done in [81]. The QT-Opt approach [82] presented in 2018 involves a number of robotic manipulations schemes and also possess the ability to deal with any disturbances dynamically. The reward scheme is simply 1 and 0 for any successful or failed grasping operation respectively. Their close-loop grasping framework is similar to the other approaches presented before [66], [83], [84]. The success rate of grasping unseen objects through this approach has been recorded at 96% after utilizing around 800 hours of 7 robots in the total time of four months.

An approach known as Grasping in the Wild [85] has also been proposed which learns from the human demonstration and performs closed-loop 6D robotic grasping on even moving objects but with some speed factor restrictions. Some other RL-based approaches [86], [87] have also been proposed which use RL framework and rewarding scheme for category level pick-and-place, meaning the geometric knowledge of the class is already supplied to the framework and then other unseen objects from same classes are grasped through learning. But these approaches lack in multiple ways such as extensive training time requirement, around 12 hours for each class and failure to recognize the actual class of the object in the clutter comprising of the unseen objects from multiple classes.

3.3 Prehensile & Non-Prehensile Robotic Manipulation

As discussed in the beginning of this chapter, there can be different criteria for categorization of robotic manipulation approaches. One of these criteria can be the types of robotic manipulations. We can divide the robotic manipulations into two categories the prehensile and the non-prehensile robotic manipulations. The prehensile manipulations are those which involve grasping and the non-prehensile manipulations are those which involve non-grasping motions such as pushing, shifting, sliding etc. In this section, we will discuss approaches focusing prehensile, non-prehensile and combination of these both robotic manipulations for the pick-and-place task.

There have been a number of grasping approaches presented previously. Some of these early approaches have been model-based and purely analytical in nature. These approaches involve modeling and measurement of contact forces and their resistance factor against externally applied wrenches [88], [89]. Whereas, some other approaches use the ability of restricting motion of an object for the ranking of grasping poses [90]. The general flow of such approaches' implementation in real-world scenarios is through using beforehand generated grasp configurations from any 3D object database [91] and afterwards ranking them with point cloud to achieve the object pose estimation [92], [93]. But as evident from the description, these approaches and techniques require object-specific knowledge such as dynamics, poses, shapes, orientations, contact points, etc. This requirement of object-specific knowledge restricts the application of these approaches in the case of unseen and novel objects in environments not known before.

In recent years, research community has focused on model-free data-driven or empirical approaches more. These approaches learn grasping policies to identify grasping opportunities on the basis of visual features instead of requiring any object-specific such as dynamics, poses, shapes, etc. [35], [38], [94]–[97]. In [97] pre-

trained models have deployed for enhancing the overall performance. These models have been trained beforehand on various other tasks such as poking etc. In [94] the utilization of FCNs has been shown for learning of policies with the help of affordances.

Like prehensile manipulation, the non-prehensile manipulation such as pushing is also studied for decades. There have been various model-based analytical approaches. Some approaches tend to model the dynamics of non-prehensile manipulation involving the forces of friction [98], [99]. Most of these modeling approaches seem strong theoretically, but they don't hold their ground in practical scenarios [100], [101]. As the friction factor is variable because it not uniformly distributed across the plane it can cause serious error in prediction through these friction modeling systems in the real-world scenarios. Like the prehensile manipulation research, in recent years some data-driven approaches have been proposed for the pushing of the objects but most of the approaches follow the principle of dealing one object at a time [102]–[104]. Tackling more complex scenarios such as clutter situation and variable friction issue in real-world systems is still a factor need to be addressed in these data-driven approaches.

Another direction of robotic manipulation can be the combination or amalgamation of the prehensile and non-prehensile robotic manipulation for the pick-and-place tasks. This area of research is definitely an interesting one but still has much room for exploration because it has not been center of focus much yet. An approach presented in [105] proposes a framework which performs push-grasping i.e. pushing while grasping the object. This benefits in two ways, first it increases the probability of successful grasping through minimizing any uncertainties with pushing and secondly it also equips the system with an additional skill of sweeping which is helpful for rearranging the objects in the clutter. A downside to this approach is that

their policies are mostly handcrafted which pushes it in the domain of analytical/geometric methods requiring object and domain-specific knowledge.

There are some model-free approaches [106], [107] too which focus on performing such pushes so the objects end up at such locations which are considered favorite for the success of pre-designed handcrafted grasping operations. These beforehand known locations are used to train the pushing policies. Now if we try to design this approach as a model-driven approach it doesn't make much sense as the policies continuously keep on changing and constant learning is achieved.

An RL-based approach proposed by Boularias et al. [108] perform training of its policies through RL so that they can choose whether to grasp or push objects in the clutter according to the manually handcrafted features. In this technique, first the state of the workplace received from the vision sensor in RGB format is segmented for identification of the objects. Once objects are identified, then multiple prehensile and non-prehensile manipulations options are sampled for each individual object. For each of these multiple manipulation options handcrafted features are generated and then the manipulation option with the highest expected reward is executed finally. One major limitation of this approach is that it is only suitable for convex objects. Another downside to this technique is that it requires beforehand handcrafted knowledge in case of non-prehensile manipulation such as the motion prediction once an object is pushed. For these predictions system is dependent on the simulator and the effect of this non-prehensile manipulation on the prehensile manipulation coming in future is also required to be known beforehand.

3.4 Summary

In the current available literature regarding pick-and-place and robotic grasping various analytical and data-driven approaches have been presented so far. In recent

years, the research community has been more focused on data-driven/empirical methods. Data-driven methods can also be called the learning-based methods because they are based on the principle of utilizing machine learning for learning the robotic grasping skills. The model-based data-driven approaches revolve around the object pose estimation function. These approaches have limitations such as pose ambiguities due to multiple object symmetries and these techniques become useless in scenarios where object-specific knowledge such as geometric knowledge is unavailable because object pose estimation require object-specific knowledge beforehand.

Therefore, the model-free robotic grasping approaches are more popular because they don't have object pose estimation function, so they don't need object-specific information. Both the discriminative and generative approaches utilize supervised learning and are considered attractive due to their ability to generalize to unseen object effectively. But these both approaches suffer from limitation of requiring large annotated datasets for training. These large datasets require a lot of hard work and months of patience for building. Moreover, any hardware changes can nullify the whole previous training effort and require fresh training on a newly collected dataset. Moreover, the discriminative approaches have also been proven not time-efficient because they require multiple forward passes of the neural network for identifying optimal grasps.

Most of the RL-based approaches also suffer from limitation of requiring manually handcrafted features during the process. Mostly these approaches focus on the objects in workspace individually which requires them utilization of techniques such as segmentation, singulation etc., thus incurring addition overhead. Any approaches which reduce the additional overhead by considering the whole workspace as a single entity, suffer from poor memory management while deploying

complex neural network architectures, ending up in a bottleneck situation and choking hardware resources such as GPU.

The following chapters of this thesis present learning frameworks which address not only these all limitations in vision-based robotic pick-and-place environments but also the non-visual robotic grasping environment learning which has not been focused before. The proposed data-driven learning-based frameworks make agents learn and perform the pick-and-place task in both vision-based and non-visual industrial setups, thus improving the efficiency and throughput of the pick and place system.

Chapter 4

4 RL-based Pick-and-Place Framework with Non-Visual Sensing

4.1 Introduction

This chapter presents a RL-based learning framework for robotic pick-and-place in non-visual industrial environment. This chapter addresses our objective of developing RL-based algorithms for learning and performing the non-visual pick-and-place tasks. The foundation laid in this chapter also provides the basis of the vision-based approaches (Chapter 5 and Chapter 6) presented in this thesis.

The industrial machine vision systems have played a very importation role in revolutionizing the industry. This evolution of vision systems led to various technological features such as full-color megapixel resolution, 3D graphical data, etc. Normally an industrial machine vision system is constituted of various modules such as lightning module, lens, capture board/sensor, processor and a communication module. Despite this advancement in industrial vision systems, still there are some scenarios which are considered not suitable for their deployment. At some occasions the high cost of the vision system becomes a hurdle whereas in some instances the installation space requirement causes the problem because only few centimeters are available at the site[3]. In some industrial setups, high vibrations or maximum dust factor also restrict the usage of vision systems[3]. In some scenarios even the washup from the water jets becomes the reason of avoiding the industrial machine vision systems[3].

All the existing learning-based approaches in the relevant literature discussed in Chapter 3 target only the vision system-based scenarios. One can witness a clearly visible research gap in the area of learning-based approaches for the non-visual scenarios. Therefore, this approach presented in this chapter, aims to develop a learning-based RL framework for the industrial pick-and-place task without involving the visions sensors. In this our designed framework, the vision sensors are replaced by the proximity sensors as our non-visual aid.

4.2 Problem Scenario

The approach presented in this chapter considers the problem of pick-and-place in a smart production line. In this smart production line, a number of variable-shaped objects are being placed and moved on a running conveyor belt in different positions and orientations. The speed of the conveyor belt can change randomly during the process. If the speed of the conveyor belt becomes fast, obviously the objects will move at faster pace and if the speed of the conveyor belt become slow, the objects will move at slower pace. At one end of the conveyor belt, which is considered as the workspace, proximity sensors are installed which detect the object upon arrival and transmit signal to the robotic arm letting it know that the object has reached the workspace for being picked-and-placed at the designated location. Once the robotic arm receives the signal from the proximity sensor, it attempts to grasp the object and place it at the designated location. In this way, we target to train and test such RL agents which can learn and perform the pick-and-place of different objects at random positions, orientations and conveyor belt speeds.

4.3 MDP Design

As this chapter proposes an approach in which the robotic arm learns to pick-and-place objects moving on the conveyor belt to the designated location. The objects moving on the belt can be at different alignment-wise positions such as left-aligned, center-aligned and right-aligned. The speed of the conveyor belt can also vary during the process such as slow speed, medium speed and fast speed. The objects under consideration are solid shapes such as cuboids, cylinders and spheres. This whole scene with variable-shaped objects at different positions can be seen in the Figure 3.

To resolve any task at hand through RL, we need to develop and design its Markov Decision Process (MDP). The five elements of a MDP have been discussed before in the Chapter 2. This our pick-and-place can be transformed into an MDP. The five key elements of our designed MDP are as follows

- **State (***s: C, N, G, speed, position, path***)** where the *C* represents any of the manually handcrafted XYZ coordinates according to the nature of the action; the *N* depicts the nature of the action which can be pre-pick, pick or place; the *G* is the potential grasping pose from the predesigned

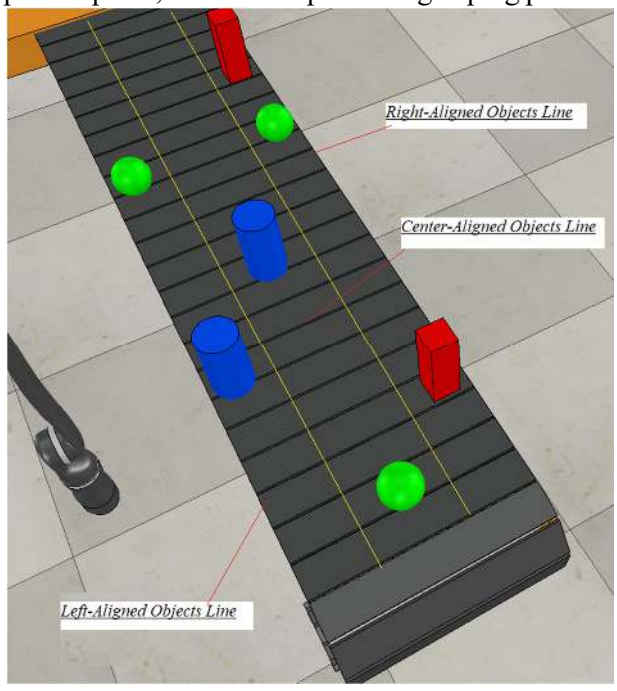

Figure 3 Conveyor belt scene

set; *speed* tells about the pace with which the object is moving on the conveyor belt i.e. slow, medium, fast; *position* represents the alignment of the object moving on the conveyor belt i.e. left-aligned, center-aligned, right-aligned; *path* represent the set of path points calculated last time from the initial position to pre-pick position, or from pre-pick to pick position, which are actually the XYZ coordinates represented by *C,* if any.

- **Action (***C, N, G, path***)** where the *C* represents any of the manually handcrafted XYZ coordinates according to the nature of the action; the *N* depicts the nature of the action which can be pre-pick, pick or place; the *G* is the potential grasping pose from the predesigned set; *path* represent the set of path points calculated last time from the initial position to pre-pick position, or from pre-pick to pick position, which are actually the XYZ coordinates represented by *C,* if any.
- **Reward (***R***)** where *R* represents the reward value. The agent is awarded reward 1 upon the accomplishment of the pick-and-place task. The agent gets 0.5 reward if the grasping is done successfully but placement has been failed. In the case of failing whole task, the grasping and placement both, 0 is given as reward to the agent. In some schemes, bonus and negatives rewards have also been tried to experiment with the learning ability of the agents.
- **Transition Function (***δ***)** where *δ* denotes the transition function which is actually a stochastic function keeping track of the transitions of the agents from state to state in the environment – but this problem is being designed as a model-free RL problem therefore here the transition function will be discovered through trial-and-error method [109].

- **Discount Factor (γ)** where γ represents the discount factor and is kept between 0 and 1 in order to increase the importance of the future rewards and make the agents learn at faster pace.

The MDP formulation of our task as stated above is then implemented through both off-policy and on-policy TD algorithms, as explained in the Chapter 2, which will be discussed in the further sections of this chapter.

4.4 General Approach

For gaining better pace of learning, we define the pick-and-place task in a modular fashion. The major part of our pick-and-place task is the grasping part. The grasping part in our approach consists of two subparts. The first subpart is to move the robotic arm gripper to the 3D coordinate near to the object, and then the other subpart is to move from this 3D coordinate to the object for grasping through following a linear path. These two sets of 3D coordinates, first which will bring the robotic arm near the objects, and second which lets the robotic arms go for grasping the object through a linear path, are manually handcrafted. For each combination of randomly varying factors like position, speed, shape, some 3D coordinates will be optimal and some will be worst.

In this our proposed approach, we divide the whole task of pick-and-place into four phases. The first phase is known as the 'Initial Phase'. The 'Initial Phase', as evident from the name, is the beginning phase where the robotic arm is waiting at standby to receive the signal from the ray-type proximity sensors installed at the workspace end of the conveyor belt. The signal transmitting from the proximity sensors not only convey the status of arrival of the object in workspace but also the alignment position and speed of the conveyor belt. Once the wait is over and proximity sensor emits the signal upon detecting the object in the workplace, the

second phase starts. The second phase is known as the ‘Pre-Pick Phase’. In this second phase, the agent selects the 3D coordinates from the first set of manually handcrafted pre-pick coordinates, and then calculates the path through motion planning module to make the end-effector reach that chosen pre-pick 3D coordinates (near to the object). Once the end-effector reaches the chosen pre-pick 3D coordinates, the approach enters the third phase known as the ‘Grasp & Pick Phase’. In this phase, with the end-effector currently at pre-pick coordinates, the agent goes forward and selects the 3D coordinates from the second set for the grasping. Once the selection is done, the linear path is calculated through the motion planning module from the pre-pick 3D coordinates to this freshly selected grasp 3D coordinates. One the end-effector reaches the grasp coordinates the gripper performs the grasping.

The visualization of the manually handcrafted two 3D coordinates sets, pre-pick and grasping, is shown in Figure 4. Each ball-like instance is actually a unique 3D XYZ coordinate. The coordinates on the left, being shown in three adjacent lines, are the potential pre-pick coordinates. The coordinates on the right, being shown in a single line, are the potential grasping coordinates. This arrangement of coordinate being shown is only for describing the concept of positions of coordinates, the real accurate positions and number of coordinates in the testbed settings may differ. Now according to the Figure 4, the agent first selects the red coordinate from the pre-pick 3D coordinates set. So, the agent calculates the path and reaches the red pre-pick coordinate. Afterwards, in the next phase, the agent selects the green grasp coordinate from the grasp 3D coordinates set, and the linear path is calculated from the red pre-pick coordinate to the green grasp coordinate and then the grasping is attempted.

After the grasping attempt, the cycle enters the last phase known as the ‘Place Phase’. In this phase, with the object inside the closed gripper, path is calculated to

the designated placement 3D coordinates and after reaching the placement location gripper is opened to place the object. And then the loop reiterates from the start again. This whole cycle of our approach consisting of four phases is shown in the Figure 5. The motion planning part involved in all the four phases will be briefed later on in this chapter.

The rewards are awarded to the agent on the basis of the success status of the grasping and the placement. The agent is awarded reward 1 upon the accomplishment of the pick-and-place task. The agent gets 0.5 reward if the grasping is done successfully but placement has been failed. In the case of failing whole task, the grasping and placement both, 0 is given as reward to the agent. In some schemes, bonus and negatives rewards have also been tried to experiment with the learning ability of the agents. These schemes allowed additional rewards for selecting the hand-marked optimal coordinates and negative rewards for failing the pick-and-place task completely. The results of these different rewarding schemes have also been compared and presented in the results section in this chapter.

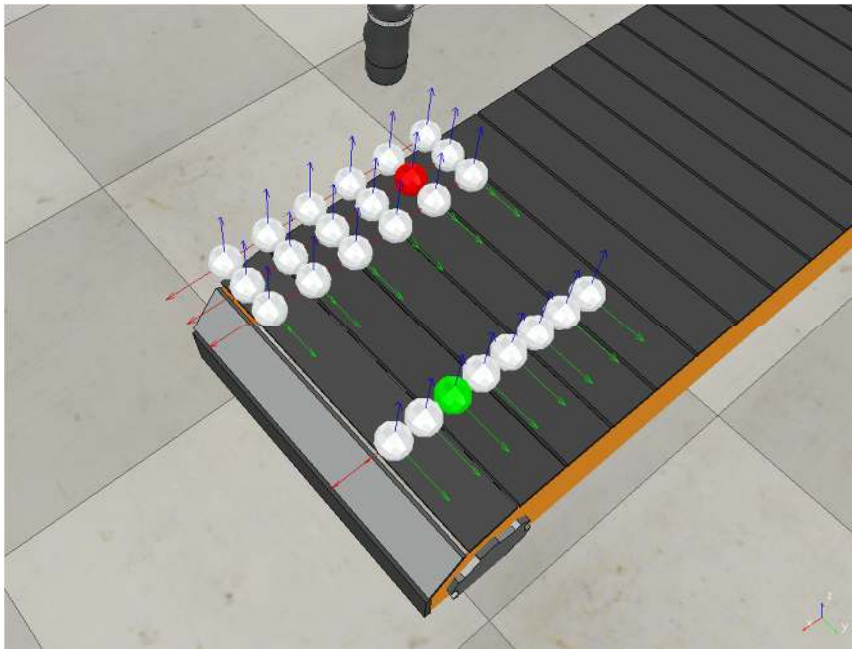

Figure 4 Pre-pick and Pick 3D coordinates

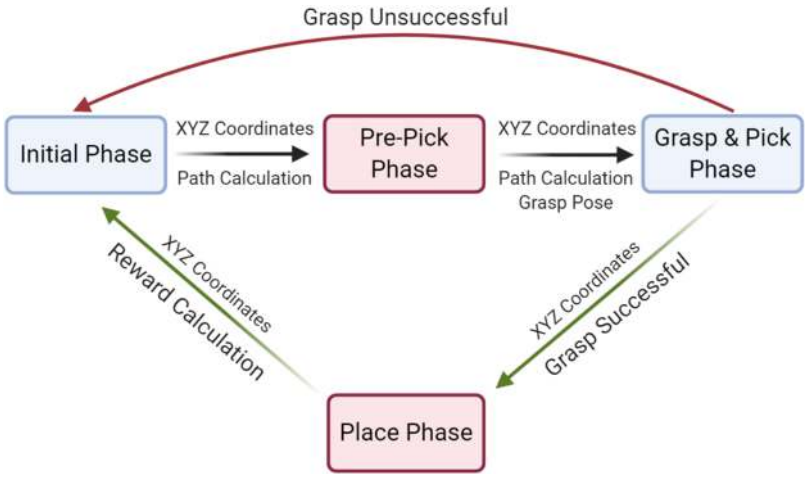

Figure 5 Phase cycle of the non-visual approach

The main idea behind designing the MDP for this episodic pick-and-place task is to make the agents learn optimal policy through trial-and-error for the maximization of the future discounted reward.

$$E_t = r_t + \lambda r_{t+1} + \lambda^2 r_{t+2} \dots \lambda^{T-1} r_T \quad (1)$$

This designed MDP is implemented, trained and tested through both the off-policy and the on-policy TD algorithms. For off-policy, Q-learning algorithm and for on-policy, SARSA algorithms have been utilized, which have been individually discussed in detail in Chapter 2. In the next section, we will briefly compare them for better understanding.

4.4.1 Q-Learning vs SARSA

This our proposed approach demonstrates the comparison of off-policy Q-learning and on-policy SARSA TD algorithms. It is important to understand the key differences between the two. In off-policy TD algorithms such as Q-learning the agent learns the optimal policy while working with two separate policies. The agent interacts with the environment according to the behavior policy but at the same time updating and evaluation is done to the estimation/target policy. Through this method, without even deploying the actual greedy policy, the Q-

learning algorithm learns to estimate expected reward and update value of expected new state [20]. On the contrary, in the on-policy TD algorithms, the behavior policy and the estimation/target policy are same, which means that the policy being used to interact with the environment is the same policy which will be updated. The agent follows the same policy throughout in order to converge [20].

The pseudocodes of Q-learning and SARSA algorithm upon examination give away the key differences between two [110]. A major difference between the Q-learning and SARSA algorithms is the way they both update the Q-values in their Q-table. In the Q-learning algorithm, the agent performs the updating by choosing the optimal action available in the new state which the agent has transitioned to. But in the SARSA algorithm, the action selection is done through following the same policy. Therefore, in SARSA algorithm, the action selection is not optimal always. If there are some unwanted or undesired states in the environment, the SARSA algorithm may end up the agent in them as the action selection is not optimal always. Due to this vulnerability, SARSA plays safely by following such a pattern which can reduce the risks of transition to an undesired state. On the other hand, the Q-learning algorithm doesn't take this possibility of landing in undesired stated into consideration, and performs the action selection completely based on the Q-values in the Q-tables which have been updated through optimal action selection always. These contradicting behaviors of the both Q-learning and SARSA algorithms can be easily witnessed by examining their application on the popular cliff-walking problem [20].

4.4.2 Motion Planning Module

Today in the field of robotics, a number of motion planner control solutions are available. For our proposed approach, different motion planning libraires and

frameworks were considered and reviewed such as Trajpot [111] and OpenRave [112]. Among all other considered option, Open Motion Planning Library (OMPL) [113] proved to be the most appropriate option due to its maximum degree of customization allowance. In OMPL, multiple control-based planners and geometric planners are available. Some of the popular sampling-based planners available in OMPL are Expansive Space Trees (EST), Single-query Bi-Direction Lazzy (SBL), Probabilistic Roadmap Method (PRM), Rapidly-exploring Random Trees (RRT), etc. The OMPL's planner deployed in our approach is a single-query planner. It is known as RRT-Connect [114], which happens to be a bi-directional variant of the Rapidly-exploring Random Tree (RRT) planner. The main idea of a RRT-Connect planner is to create two RRTs, where one is located at the start point and the other one is located at the end point, and then connect both. Utilizing two RRTs simultaneously is the reason behind its ability to outperform the basic RRT planner.

4.4.3 Simulation Environment

We have discussed a number of physics engine-based robotic simulators in detail in the Chapter 2. Every simulator has its own advantages and disadvantages and it may be suitable for particular tasks only. According to our task requirement and resources, the Virtual Robot Experimentation Platform (V-REP) proved to be the most suitable one. The V-REP is actually a 3D robotic simulator which involves the integrated support for development and coding [30]. It is equipped with multiple physics engines such as ODE and Bullet variants in order to achieve the real-time emulation of all the objects during the simulations. The availability of multiple APIs and the functionalities such as threaded/non-threaded Lua scripting enables the V-REP to develop cross-platform application through combining platforms such as Python, C++, Java, etc. With the help of the python-

based API, it was possible for us to create a python-based RL agent which was able to interact with the environment inside the V-REP being controlled through the Lua scripts.

V-REP also provides the Forward and Inverse Kinematics calculation module. In Forward kinematics, joint parameters are supplied as input to the kinematics equations and the position of the end-effector is calculated as the output. On the contrary, in the Inverse kinematics, the required position of the end-effector is supplied as input and the joint parameters are calculated as the output [115]. Another importation module of V-REP utilized in this approach is the collision-detection and collision-avoidance module which detects and avoids any possible collision of the robotic arms with the environment. In our experimentation trials, we used the JACO robotic arm which is a 6 degree-of-freedom (DOF) robotic arm. For the grasping part, we attached a RG2 gripper to our JACO robotic arm. The grasping poses of our JACO robotic arm with RG2 gripper from our approach are shown in the Figure 6.

4.5 Experimental Results & Discussion

We trained and tested both RL agents, one with the off-policy Q-learning TD algorithm implementation and the other with on-policy SARSA TD algorithm in our designed environment in V-REP simulator. Our experimental trials of training and testing are divided into two categories for better and clear assessment of the learning abilities and performance of our both Q-learning and SARSA agents. These categories are explained in detail in next sections.

4.5.1 Individual Object Category Training & Testing

In this category of experimentation, we trained and tested the learning performance of both Q-learning and SARSA agents separately for all three object

shapes we described earlier i.e., cuboid, cylinder and sphere. For more complex training and evaluation these individual categories were combined too which will be explained in the next section. In this section we discuss the performance of the both RL agents over individual object category.

For each object category, we trained and evaluated the performance of both Q-learning and SARSA agents for each of the object alignment settings and the conveyor belt speed settings separately. As described in previous sections, we designed three object alignment settings i.e., left-aligned, center-aligned and right aligned, and also three conveyor belt speed settings i.e., slow, medium and fast, in our framework. Table I shows the success rate of our off-policy Q-learning agent's pick-and-place for each object category at each alignment option at a slow speed conveyor belt. In the same manner, Table II and III show the success rate of our off-policy Q-learning agent's pick-and-place for each object category at each alignment option at medium and fast speed conveyor belts respectively.

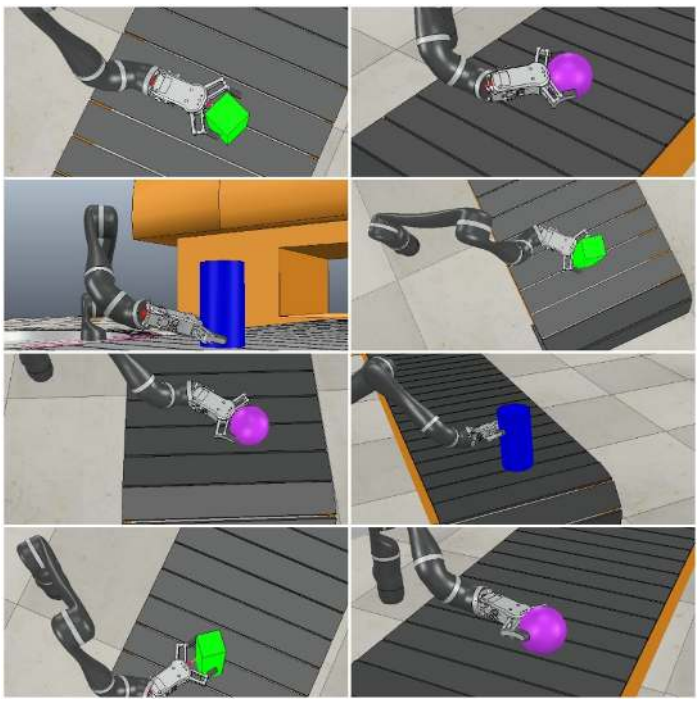

Figure 6 Grasping poses of JACO robotic arm

Table II Q-Learning Agent's Success Rate (%) At Slow Speed

Object Category	Training Performance			Testing Performance		
	Left	***Center***	***Right***	***Left***	***Center***	***Right***
Cuboid	89%	95%	91%	95%	99%	96%
Cylinder	92%	91%	88%	94%	93%	93%
Sphere	83%	88%	85%	92%	95%	91%

Table III Q-Learning Agent's Success Rate (%) At Medium Speed

Object Category	Training Performance			Testing Performance		
	Left	***Center***	***Right***	***Left***	***Center***	***Right***
Cuboid	91%	93%	93%	96%	98%	96%
Cylinder	89%	94%	90%	96%	99%	97%
Sphere	84%	90%	87%	93%	94%	94%

Table IV Q-Learning Agent's Success Rate (%) At Fast Speed

Object Category	Training Performance			Testing Performance		
	Left	***Center***	***Right***	***Left***	***Center***	***Right***
Cuboid	91%	95%	96%	97%	99%	95%
Cylinder	93%	91%	90%	95%	95%	96%
Sphere	78%	83%	85%	90%	91%	93%

The testing results shown in Table I, II and III clearly indicate that the Q-learning agent has successfully exercised its ability to learn the optimal 3D coordinates for each object position alignment and conveyor belt speed option in all three object categories. Multiple inferences and conclusions can be drawn from these results such as that the Q-learning agent has struggled more in the learning of the pick-and-place task for the sphere object category as compared to other two categories. This struggle is more visible in fast speed conveyor belt results but remains true for all speed options.

Another observation which can be made seeing the testing results is that the Q-learning agent has shown its best performance for all the object categories when they were center-aligned at medium speed as compared to other position and speed options. This means that the agent was most successful in identifying and learning

the optimal and accurate 3D coordinates and grasping poses for all center-aligned objects coming at medium speed. It clearly shows the level of efficient learning achieved by the Q-learning agent in the training phase.

Table IV, V and VI show the success rate of our on-policy SARSA agent's pick-and-place for each object category at each alignment option at slow, medium and fast speed conveyor belts respectively.

Table V SARSA Agent's Success Rate (%) At Slow Speed

Object Category	Training Performance			Testing Performance		
	Left	***Center***	***Right***	***Left***	***Center***	***Right***
Cuboid	83%	81%	80%	82%	82%	80%
Cylinder	82%	80%	78%	80%	81%	78%
Sphere	80%	79%	79%	78%	79%	77%

Table VI SARSA Agent's Success Rate (%) At Medium Speed

Object Category	Training Performance			Testing Performance		
	Left	***Center***	***Right***	***Left***	***Center***	***Right***
Cuboid	81%	78%	75%	79%	77%	73%
Cylinder	80%	78%	76%	78%	80%	77%
Sphere	77%	77%	73%	78%	77%	72%

Table VII SARSA Agent's Success Rate (%) At Fast Speed

Object Category	Training Performance			Testing Performance		
	Left	***Center***	***Right***	***Left***	***Center***	***Right***
Cuboid	79%	78%	72%	78%	78%	71%
Cylinder	77%	78%	73%	78%	77%	72%
Sphere	75%	72%	71%	76%	71%	69%

These results confirm that our SARSA agent has also managed to gain learning of 3D coordinates for each object position alignment and conveyor belt speed option in all three object categories. However, the overall performance results of our SARSA agent have been lower as compared to our Q-learning agent results. A performance deterioration trend can also be seen in the SARSA agent results as the speed of the conveyor belt increases. This down trend means that the lacking and inefficiency in the learning of the optimal 3D coordinates of the agent was directly proportional to the magnitude of the conveyor belt's speed. As much speed

increased that much learning decreased. Same as the Q-learning agent, the SARSA agent also suffered in performance with the spherical objects. The results also determine that SARSA agent did more justice to left-aligned and center-aligned objects at all the speeds as compared to the right-aligned objects.

4.5.2 Random Objects Training & Testing

In the previous section, we saw training and testing performance of all object categories individually at all position alignment and speed options separately. In this section, we take a step ahead and do performance evaluation of both Q-learning and SARSA agents by running episodes of random shape objects at random alignment positions and random conveyor belt speeds. From the training perspective, this random setting means that in first iteration of the episode agent has to pick-and-place a center-aligned cuboid object at medium speed, but in the very next iteration there could a right-aligned spherical object at fast speed and in the third iteration completely random options again. Both Q-learning and SARSA agents once trained in this random setting manner, are evaluated through specifically designed 5 test cases separately with 50 iterations in each. In test case 1, random objects at random speed are managed but it is made sure that each alignment position gets around one third of these random objects. In test case 2, random objects at random alignment positions are produced but it is made sure that each speed option gets around one third of these random objects. In test cases 3, 4 and 5 we keep the alignment positions and conveyor belt speeds completely random but at least 40% of the objects are cuboid, sphere and cylinder respectively. The testing view can be seen in the Figure 7. Table VII shows the performance results of these abovementioned test cases.

Table VIII Random Test-Cases Success Rate (%)

Test Case	Q-Learning Agent's Success Rate	SARSA Agent's Success Rate
Test Case 1	93%	82%

Test Case	Q-Learning Agent's Success Rate	SARSA Agent's Success Rate
Test Case 2	95%	81%
Test Case 3	99%	80%
Test Case 4	83%	77%
Test Case 5	97%	81%

According to the random testing results, the average success rate of our Q-learning agent is around 93%. The Q-learning agent managed to perform well in all test cases except the test case 4. The agent struggled to perform in the test case 4 as compared to other test cases because this test case contained large portion of spherical objects i.e., more than 40%. This lack in performance was also visible in the results of sphere objects category in the individual object categories testing. On the other hand, the SARSA agent's average success rate was recorded around 80% and its least scoring test case was also test case 4 like the Q-learning agent. In overall performance

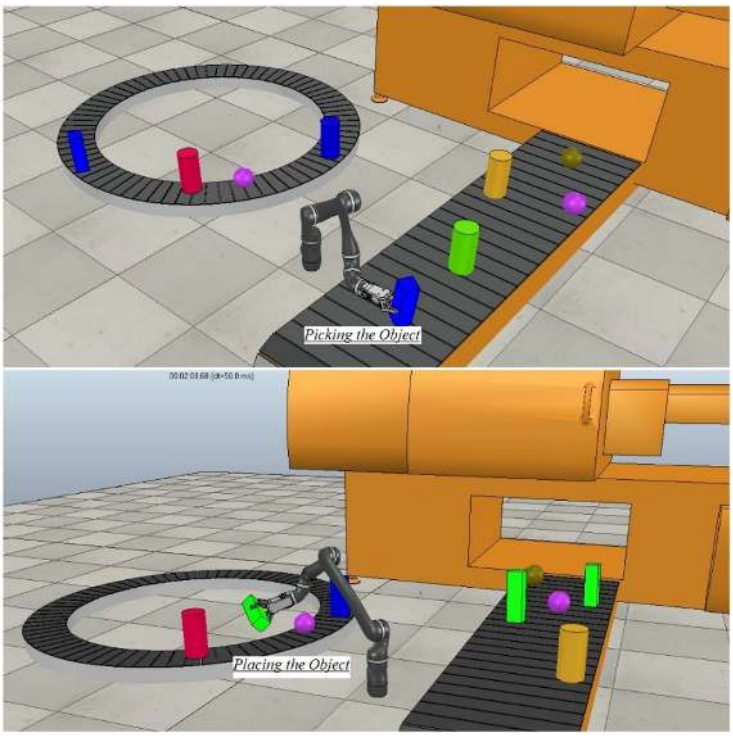

Figure 7 Testing Scene of non-visual approach

comparison, it is clearly evident that the Q-learning agent outperformed the SARSA agent in the random testing too. SARSA was outperformed because of different reasons such as its failure to perform at higher speeds of the conveyor belt and failing to learn optimal 3D coordinates for the right-aligned objects.

We have also compared the training performance of the Q-learning and SARSA agents. As described above, in this random training the agents experience the pick-and-place of random shape objects at random alignment positions and random conveyor belt speeds. In the Figure 8 the success rate of both agents is plotted against the total training steps, which are 3000. The plotted graph clearly shows that the Q-learning agent, in orange, completely outperformed the SARSA agent in blue. In the beginning, SARSA agent seemed better performing than Q-learning agent. Maybe it was because SARSA agent was preferring exploitation over exploration at that time but afterwards the scenario completely changed.

4.5.3 Reward Scheme Variations

As we discussed previously in this chapter, the rewards are awarded to the agent on the basis of the success status of the grasping and the placement. The agent is awarded reward 1 upon the accomplishment of the pick-and-place task. The agent

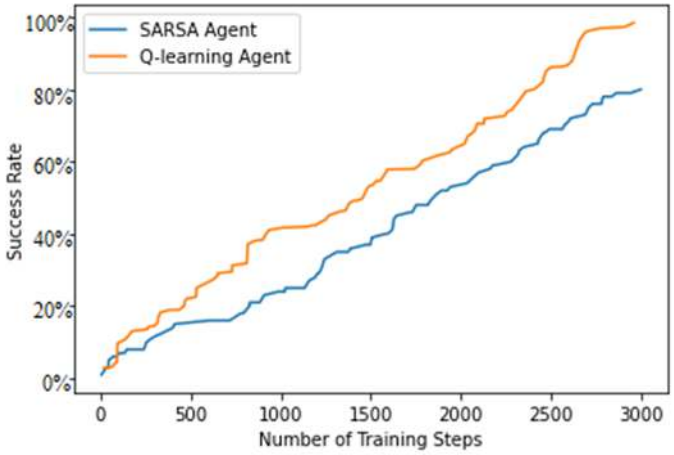

Figure 8 Performance comparison of SARSA and Q-learning agents

gets 0.5 reward if the grasping is done successfully but placement has been failed. In the case of failing whole task, the grasping and placement both, 0 is given as reward to the agent.

In order to enhance the learning abilities of agent and witness the role of rewarding scheme, we also trained variants of Q-learning and SARSA agents. These variants implement negative reinforcement learning and bonus rewarding. In negative reinforcement learning, the agent is awarded -1 instead of 0 upon failing the pick-and-place task completely and in bonus rewards, the agent is awarded an additional reward of 1 for choosing some selective manually identified optimal 3D coordinates. Figure 9 and Figure 10 present the performance comparison of the Q-learning agent and SARSA agent with normal reward scheme and their variants with negative and bonus reward scheme respectively when trained for the same number of training steps. In both graphs, the blue line depicts the performance of normal reward scheme and the orange line presents the performance of the negative reinforcement learning and bonus reward scheme. In the Q-learning agent's case, the negative and bonus reward scheme variant has performed better than the normal reward scheme. But on the contrary, the SARSA agent's both variants, normal one

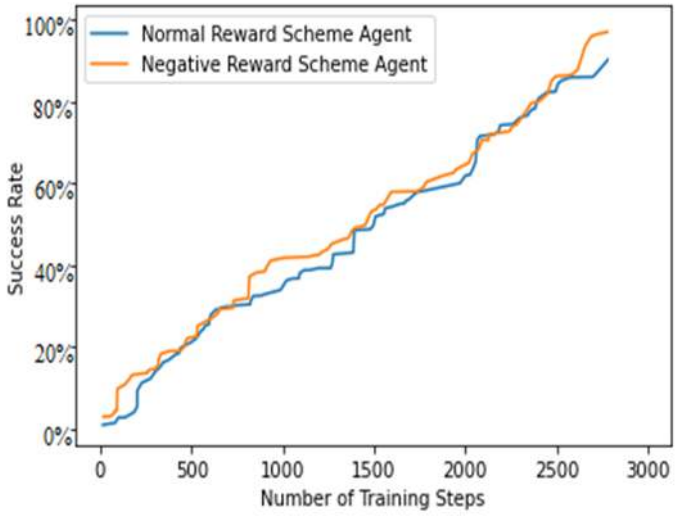

Figure 9 Performance comparison of Q-learning normal reward scheme agent and negative reward scheme agent

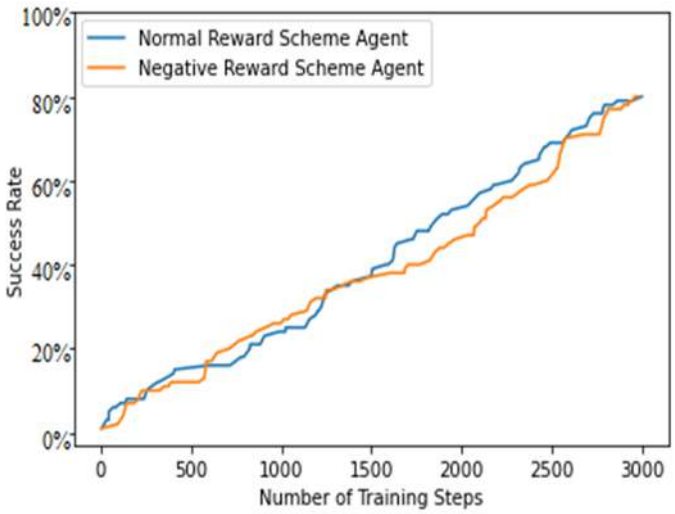

Figure 10 Performance comparison of SARSA normal reward scheme agent and negative reward scheme agent

and negative one, have not much difference in overall performance. This analysis tells us that Q-learning agent's learning ability was increased when it was introduced to negative and bonus reward scheme, but SARSA agent's lacking as compared to Q-learning agent are far enough to be covered by negative or bonus rewards.

4.5.4 Discussion

We designed the testbed in the simulated environment and then trained and tested both RL agents, the Q-learning agent and the SARSA agent. Both agents were trained and evaluated in terms of performance for individual object categories pick-and-place and random object categories pick-and-place. In both scenarios, the Q-learning agent performed better as compared to the SARSA agent.

The SARSA agent has been seen struggling with the pick-and-place of the objects as the conveyor belt increased, leading to performance deterioration. Such downfall in performance has also been witnessed in the case of right-aligned objects in both individual object categories trials and random objects trials. The spherical

objects have been difficult for both of our RL agents but it is evident from the results that Q-learning agent dealt better with them as compared to the SARSA agent.

The reason behind the lacking in the SARSA agent's performance as compared to the Q-learning agent can be the exploration-exploitation dilemma. As we know, the SARA agent always tends to follow a safer path in order to avoid any such event which can make it earn a very bad reward and transition to an undesired state. It may seem like a reasonable and logical approach but at the same time it has flaws too. This behavior makes the agent conservative and highly minimizes its intent to explore and discover any new optimal paths. But on the contrary, the Q-learning agent behaves differently. It doesn't worry much about the consequences such as landing in an undesired state. It performs the action selection completely based on the Q-values of the Q-table and go for the optimal one. This difference of behavior has played vital role in the better performance and optimal convergence of the Q-learning agent as compared to the SARSA agent.

4.6 Summary

This chapter presents an approach to address the problem of industrial pick-and-place in a non-visual environment. There are various industry setups having a non-visual environment means they don't have vision sensors installed in their setup. There can be different reasons behind restricting the vision sensors such as cost, space, vibrations, dust, wash up, etc. In our approach, we fill the gap of vision sensors with the ray-type proximity sensors. We formulated the pick-and-place problem as a MDP and solved it through a RL-based framework. We deployed both Q-learning off-policy TD algorithm and SARSA on-policy TD algorithm in the designed framework and

compared their performance. Both agents learnt pick-and-place of different shape objects at variable alignments positions and speeds.

Our performance evaluation on random objects at random alignment positions and speeds showed that the average success rate of Q-learning agent and SARSA agent around 93% and 80% respectively. We also witnessed performance increase in the Q-learning agent upon negative and bonus reward schemes application.

There are also some known limitations in the presented approach such as long training time requirements. On an Intel Core i7-3770 @ 3.40 GHz machine with 12 GB installed RAM it took around 4 to 5 hours for training for each object class. Geometric complexities of some objects such as spherical objects also become hurdle as no prior geometric knowledge is used in this approach and proximity sensors are used instead of the vision sensors. In the next chapter, we will develop the approach based on vision sensor in order to address these geometric limitations in this current non-visual approach.

Chapter 5

5 DRL-based Pick-and-Place in Clutter using Vision Sensors

5.1 Introduction

This chapter presents a DRL-based learning framework for robotic pick-and-place of regular and irregular-shaped objects with the help of vision sensors in an industrial environment. This chapter addresses our objective of developing DRL-based algorithms for learning and performing the vision-based pick-and-place tasks. The foundation laid in this chapter also provides the basis of the vision-based multiple robotic manipulation (Chapter 6) in this thesis.

Due to recent advances in last years in the field of machine vision, the vision-based industrial robotic manipulations have been the focus of the researchers. Usually, the terms computers vision and machine vision are used interchangeably but they differ from each other in a sense. In computer vision, usually there is a computer processing data after digitizing an image and then performing some action accordingly. But the machine vision is purely industrial application of the computer's ability to see and perceive. As this research area have been under focus of the research community from last few years, a number of approaches have been proposed which have been discussed earlier in the Chapter 3. In the literature review earlier, we reviewed various learning-based data-driven vision-based robotic grasping and pick-and-place approaches. Some of those approaches were based on the idea of object pose estimation, which were limited by requiring geometric object-specific knowledge beforehand. Therefore, such approaches are known as model-based approaches. We also reviewed the model-free approaches using supervised learning.

The novelty of the proposed approach in this chapter is as follows:

- Unlike other mostly existing relevant approaches, the proposed approach considers the whole workspace as one instead of considering each object individually. Therefore, it doesn't require any additional steps causing any additional overhead such as segmentation or singulation for the identification of individual objects in the clutter.
- The proposed approach requires no geometric object-specific knowledge beforehand, unlike most of the existing approaches in the literature.

The promising training and testing results of this approach and its comparison with its some variants are also presented in this chapter.

5.2 Problem Scenario

The approach presented in this chapter considers the problem of pick-and-place with the help vision sensors in a smart production line. In this smart production line, a number of regular and irregular-shaped objects are being produced in a clutter and moved on a running conveyor belt. In this clutter we make sure that there are a number loosely packed regular and irregular-shaped object at random positions and orientations so that there should be reasonable grasping opportunity available for every object. At one end of the conveyor belt, where the robotic arm is installed for the pick-and-place operation, is our workspace. The workspace is monitored by the installed vision-sensors from the top angle. Once the random loosely packed clutter of regular and irregular-shaped objects reaches the workspace, and enters the field of view of the vision sensors, the conveyor belts stop for the pick-and-place operation. The vision sensors capture the scene of the workspace and feed it to our designed framework for the processing which will be explained in detail in the next

sections of this chapter. Our framework outputs the optimal location for grasping in the workplace to the robotic arm. As soon as the robotic arm, receives the signal, it goes for the grasping and then placement at the designated location, which is a basket in this case. Rewards are awarded and the cycle continues. In this way the robotic arm, clears the whole clutter and then the next clutter arrives for pick-and-place operation. In this way, this loop of pick-and-place operation continues.

5.3 MDP Design

As described earlier, in this proposed approach we present a self-learning framework for the pick-and-place of regular and irregular-shaped objects in clutter from the workspace area of the conveyor belt under the monitoring of the vision sensors. The RL agent is trained to grasp the objects and place them into the bin. The ability to learn is ensured by the deployment of the learning algorithms of the RL family.

In order to resolve the problem as a RL problem, the first need is to design the Markov Decision Process (MDP) of the task at hand. Generally, in any MDP, initially the agent can be at a state s_t at any timestep t. As the agent follows a policy π, the agent can transition to the next state s_{t+1} after taking an action a_t, meanwhile earning the reward $r_{at}(s_t, s_{t+1})$ and the cycle continues. The main goal behind this whole continuous process is to discover and learn such an optimal policy π^* which can ensure that maximum cumulative reward is scored in the future. The MDP designed for this approach which will be explained later in this section, is resolved through the model-free off-policy TD algorithm Q-learning. The Q-learning behaves as a greedy algorithm because it goes for highest action-value for the selection of action to be performed. An action-value can be defined as a measure of expected reward to be earned in the future if a certain action is taken in a certain

state. These action-value are also known as Q-values and are calculated through the Q-function. A Q-function can be depicted as $Q(s_t, a_t)$ where agent is currently at state s_t and chooses the action a_t to be performed at any timestep t. The Q-function approximation can be explained through the Equation 1 as follows

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{at}(s_t, s_{t+1}) + \gamma Q\left(s_{t+1}, argmax(Q(s_{t+1}, a_t)) - Q(s_t, a_t)\right)\right] \quad (1)$$

Where γ represents the discount factor which tells the agent to whether prefer the learning of the immediate rewards or the distant rewards in the future by discounting the earned rewards. The α denotes the learning rate or the step size of the algorithm, which determines the weight of the more recent experience as compared to previous ones.

Our designed MDP for this approach consists of the following major elements:

- **States**: In the proposed approach, vision sensors have been used for the pick-and-place task. Through these vision sensors, the current view of the workspace can be perceived. An area of 224x224 pixels on one end of the conveyor belt is designated as the workplace. The objects in clutter are grasped from this workspace and placed into the bin. We deploy two types of the vision sensors known as orthographic visions sensor and perspective vision sensor. These both vision sensors are different from each other only because of their field of view. The orthographic vision sensors have rectangular field of view and the perspective vision sensor have trapezoidal field of view. The location of installed sensors and their field of views can be seen in the Figure 11. Both of these vision sensors gather RGB-D images of the workspace and then these images are merged together to increase the RGB and depth details. An RGB-D images consists of two pars, the RGB part which stores the color-based information through three channels red, green, blue (RGB), and the D part stores the depth information through only one channel, which we call as

height-from-bottom channel (D). For the depth part of the RGB-D image each pixel is presented in relation to distance between the object in foreground and the background [116]. Once the RGB-D image showing the current state of the workspace is ready, respective RGB-D heightmaps are generated. RGB-D heightmaps are generated by following the technique presented in [94], where firstly the RGB-D data is projected onto a 3D point cloud and then it is back-projected upwards orthographically in the gravity direction. In this manner RGB-D heightmaps are generated using both the color (RGB) and depth (D) information. Each RGB-D heightmap is actually our state s_t at any timestep t. The RGB-D images used for the heightmap generation were of the workspace which is 224x224 pixel size, therefore the heightmaps or states are also of same size. Each state is actually composed of 50,176 individual pixels, where each pixel represents a unique 3D location of the workspace.

- **Actions**: This approach is designed for the pick-and-pace task, therefore we simply design two main actions, the pick and the place action. The Pick action is the action where the robotic arm gripper grasps an object, which is the actual task that needs to be learned. In each state s_t at any timestep t, which is a heightmap, a particular pixel p will be selected which have a corresponding 3D location p' in the workspace as described in Equation 2.

$$p \rightarrow p' \in s_t \qquad (2)$$

This p' is the location where the pick action will be performed by the robotic arm. This p' is considered as the mid-point for grasping by the

parallel-jaw gripper of the robotic arm. The agent automatically adds 3 cm to the Z-axis component of the 3D location *p'*, and then perform grasping. The other action, place action, is performed once the grasping is done successfully. The grasped objects are placed into the bin through this action. Motion planning and path calculation for these actions are done on the runtime and saved for any future reuse Motion planning module will be seen in a further section.

- **Transition Function:** As we know, the transition function is responsible for keeping the transition records from one state to another. Our designed approach is completely model-free therefore it has no predetermined transition table, instead it is learnt through the experience. As we deploy the off-policy Q-learning TD algorithm, the Q-function is to be approximated to gain the Q-values. The Q-values, also known as action-values are actually the prediction of expected rewards in the future. For Q-function approximation we utilize the deep

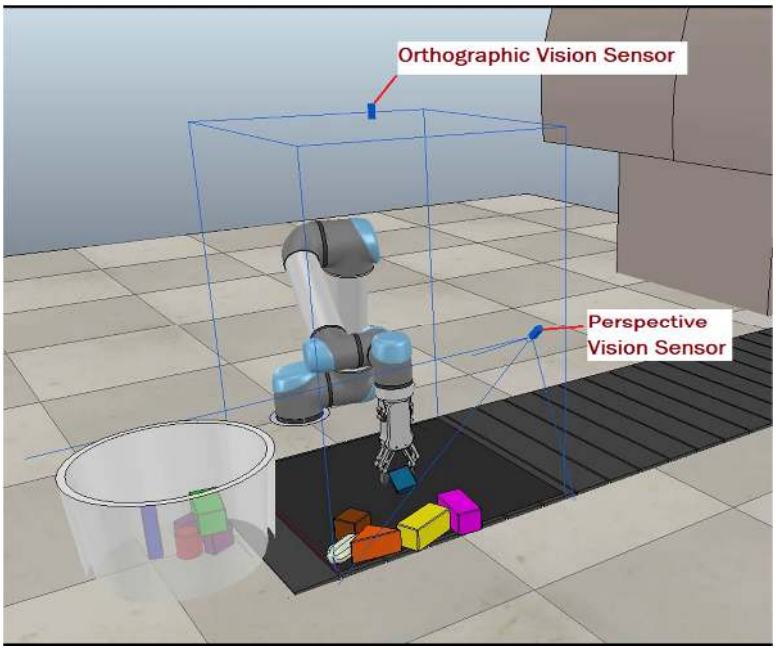

Figure 11 Vision sensors installation in the simulation

Q-networks [5], [23]. This pixelwise parameterization technique has also been utilized before in [94], but it was designed as a deep learning based approach for pixelwise predictions in supervised manner using manually annotated datasets of RGB-D images, whereas we propose a RL-based self-supervised (with the help of rewards) framework in this chapter. The Q-function in our approach is an FCN, which is supplied with the state (heightmap). It processes the state and outputs a pixelwise map of the Q-values. As the state is a RGB-D heightmap of size 224x224 pixels, the pixelwise map generated as output is also of the same size. This means that each pixel of the state has a corresponding Q-value in output pixelwise map. In other words, the corresponding Q-value from the pixelwise map predicts the expected future reward if the action is performed at that pixel *p*'s corresponding 3D location *p'* in the workspace as shown in Equation 3.

$$argmax\ Q(s_t, a_t) = argmax(FCN_{grasp}(s_t),) \quad (3)$$

The FCN, being used as the Q-function, utilizes the architecture known as DenseNet-121[117]. We worked with its both variants, the pretrained and untrained. The factor which differentiates the DenseNet from other architectures of FCNs is the nature of links between its layers. Usually, a general FCN having *L* layers has *L* links but the DenseNet with *L* layers have *L(L+1)/2* links among the layers. This enhanced connectivity among the layers in the DenseNet tackles multiple problems like vanishing gradient and other issues relevant to feature reuse and propagation. We deploy the basic architecture of Dense-121 and extend it by adding two 1x1 layers along with ReLU activation and batch normalization. Two trunks are used, where one trunk is supplied

with the color data and the other trunk is supplied with the depth data. Moreover, a rotation function is also applied to RGB-D heightmaps before feeding to the FCN so that the degree of learning can be increased and multiple orientations can help in the process. The detail of this rotation function will be discussed in the Chapter 6.

- **Rewards:** In this approach, we use the reward scheme as the one in non-visual approach given in Chapter 4 where the agent is awarded reward 1 upon the accomplishment of the pick-and-place task. The agent gets 0.5 reward if the grasping is done successfully but placement into the bin has failed. In the case of failing whole task, the grasping and placement both, 0 is given as reward to the agent. A failed grasp and placement are identified through a proximity sensor installed between the robotic gripper and in the bin respectively.
- **Discount Factor:** The discount factor and is kept between 0 and 1 in order to increase the importance of the future rewards and make the agents learn at faster pace.

5.4 Training Specifications

In this proposed scheme, the target is to make the agent learn the pick-and-place of the regular and irregular-shaped objects in a clutter on the conveyor belt once they reach the workspace area with the help of the orthographic and perspective vision sensors. The clutter size is kept between 10 to 15 regular and irregular-shaped objects. It is ensured that the clutter is loosely packed so that there should be reasonable grasping opportunity available for every object. Because this approach aims to prove that pixelwise-parameterization technique works efficiently in the process of learning, therefore initially complexity is reduced by keeping the clutter loosely packed. We extend this framework for increased complexity situations in the Chapter 6.

Once the loosely-packed clutter of 10 to 15 regular and irregular-shaped objects reach the workspace, the vision sensors capture the RGB-D state of the workspace. This RGB-D state is converted to RGB-D heightmaps which are fed to the FCN as input for the Q-function approximation. The FCN process the input through its all layers of the network and as result outputs a pixelwise map of the action-values or Q-values. As we have deployed the off-policy Q-learning algorithm to resolve this task, the Q-learning goes for the most optimal option during the action selection process. Therefore, the pixel *p* with the highest corresponding action-value in the pixelwise map is selected for the grasping action. The grasp is attempted by the robotic arm gripper at the corresponding 3D location *p'* to the chosen pixel *p*. The z component is automatically programmed to be incremented by 3 cm in order to make the grasp probability higher. Once the successful grasp is made, decided on the basis of the proximity sensor installed between the gripper jaws, the robotic arms move towards placing it in the designated location which is the bin. This practice is continued until all workspace is cleared and all the objects have been placed into the bin.

This one complete cycle, as shown in Figure 12, of clearing the workspace by picking and placing all the objects is considered as an epoch. Such 3000 epochs are completed while training the agent. In each epoch, the clutter is designed randomly. In case of agent failing continuously for more than certain number of attempts in grasping an object the epoch is restarted to start from scratch again.

In our FCN designed using the DenseNet-121 architecture, for the backpropagation part, Huber Loss function [118] is used as our loss function and the stochastic gradient descent (SGD) [119] is utilized as the optimizer. The important aspect of backpropagation in our approach to understand is that the pixel *p* selected out of total 50,176 pixels for performing action due to its highest action-value in the generated pixelwise map is the only pixel for

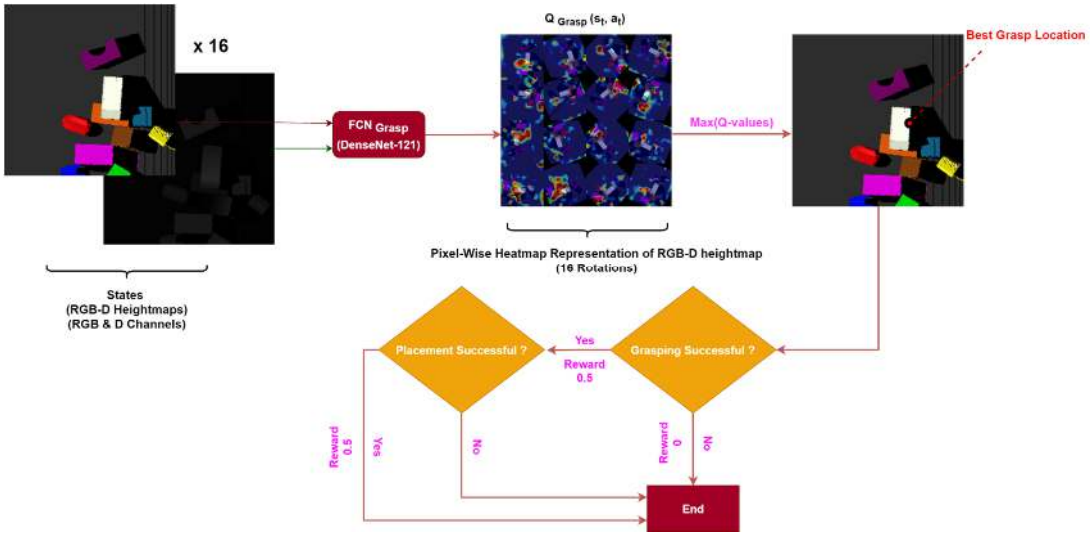

Figure 12 Flow of vision-based pick-and-place approach

which the gradient passing procedure is done. The remaining 50,175 pixels experience zero loss backpropagation.

5.4.1 Motion Planning Module

Today in the field of robotics, a number of motion planner control solutions are available. For our proposed approach, different motion planning libraires and frameworks were considered and reviewed such as Trajpot [111] and OpenRave [112]. Among all other considered option, Open Motion Planning Library (OMPL) [113] proved to be the most appropriate option due to its maximum degree of customization allowance. In OMPL, multiple control-based planners and geometric planners are available. Some of the popular sampling-based planners available in OMPL are Expansive Space Trees (EST), Single-query Bi-Direction Lazzy (SBL), Probabilistic Roadmap Method (PRM), Rapidly-exploring Random Trees (RRT), etc. The OMPL's planner deployed in our approach is a single-query planner. It is known as RRT-Connect [114], which happens to be a bi-directional variant of the Rapidly-exploring Random Tree (RRT) planner. The main idea of the RRT-Connect planner is to create two RRTs, where one is located at the start point and the other one is located at the end point, and then connect both. Utilizing

two RRTs simultaneously is the reason behind its ability to outperform the basic RRT planner.

5.4.2 Simulation Environment

According to our task requirement and resources, the Virtual Robot Experimentation Platform (V-REP) proved to be the most suitable physics engine-based simulator. The V-REP is actually a 3D robotic simulator which involves the integrated support for development and coding [30]. It is equipped with multiple physics engines such as ODE and Bullet variants in order to achieve the real-time emulation of all the objects during the simulations. The availability of multiple APIs and the functionalities such as threaded/non-threaded Lua scripting enables the V-REP to develop cross-platform application through combining platforms such as Python, C++, Java, etc. With the help of the python-based API, it was possible for us to create a python-based RL agent which was able to interact with the environment inside the V-REP being controlled through the Lua scripts.

V-REP also provides the Forward and Inverse Kinematics calculation module. In Forward kinematics, joint parameters are supplied as input to the kinematics equations and the position of the end-effector is calculated as the output. On the contrary, in the Inverse kinematics, the required position of the end-effector is supplied as input and the joint parameters are calculated as the output [115]. Another importation module of V-REP utilized in this approach is the collision-detection and collision-avoidance module which detects and avoids any possible collision of the robotic arms with the environment. In our experimentation trials, we used the famous UR5 robotic arm of Universal Robotics which is a 6 degree-of-freedom (DOF) robotic arm. For the grasping part, we attached a RG2 gripper to our UR5 robotic arm.

5.5 Experimental Results & Discussion

In this section we see the performance of our self-learning framework in the pick-and-place task with the help of vision sensors in training and testing phases.

We present the performance results of number of experimental trials conducted to evaluate the learning capability of the RL agent through our self-learning vision-based pick-and-place framework.

In order to compare the performance, we first set a baseline. For the baseline, we create another RL agent with same pixelwise parameterization scheme but switch the Q-function's architecture to pretrained ResNet-101 instead of the DenseNet. Only the Q-function approximator's architecture is changed, other elements in the scheme such as states, actions and reward scheme remain the same. ResNet is the abbreviation of the Residual Network. The residual network was primarily designed to tackle the vanishing gradient problem, which it accomplishes with the help of the skip connections. Our proposed approach is named as PnP approach and the baseline approach based on ResNet is named the ResNet-based approach. These both agents were trained according to the training specifications mentioned in previous sections. Both agents were trained for 3000 epochs. The Figure 13 shows the performance graph of both agents while training. Our PnP approach, in red line, outperforms the baseline approach ResNet-based approach, in orange line, around by 16%. The suspected possible reason behind this performance margin between the two agents is feature learning mechanism difference. As the ResNet architecture is designed to apply feature summation, but the DenseNet architecture performs feature concatenation with the help of the $L(L+1)/2$ links among the L layers. This difference of summation and concatenation enables the DenseNet-based approach, the PnP approach, perform better due to the gain and enhancement through the feature reusability.

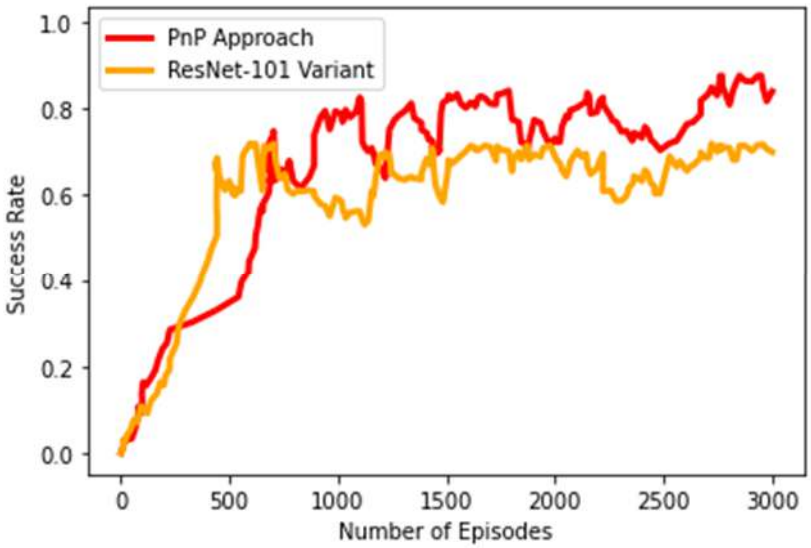

Figure 13 Performance comparison of our PnP approach and ResNet-based approach

In order to get more insight of the learning process, we trained another two variants of our PnP approach. When we discussed the transition function of the designed MDP in the earlier section, we mentioned the utilization of the DenseNet-121 architecture in both pretrained and untrained formats. The DenseNet-121 architecture being used as Q-function approximator in our PnP approach is the pre-trained variant. This pretrained variant has been trained on the ImageNet dataset. So, in order to evaluate the roles of weights from the ImageNet dataset training, we created the other variant which is to be trained from the scratch, named as No Pretrained Weights variant.

Another important element in our PnP approach is the depth data (D) which we gather through our orthographic and perspective vision sensors along with the color data (RGB). This depth seems important and crucial to the learning because it contains the height-from-bottom information which is actually required to differentiate the background from the object in the foreground through the distance factor. So, for evaluating the role of depth data in the learning process, we designed a variant where the agent is similar to the PnP agent, but it never gets any depth data (D) but only color data (RGB). We named this variant as No Depth Channel variant.

These both variants, the No Pretrained Weights variant and the No Depth Channel variant were trained in the same manner as the PnP approach for 3000 epochs. The Figure 14 shows the performance comparison of these two variants to the PnP approach. The red line shows the performance of the PnP approach and the blue line shows the performance of the No Pretrained Weights variant. It is clearly evident that there is not much difference between the success rates of these two. In fact, the early rise of the No Pretrained Weights variant shows that the pretrained weights in the PnP approach are causing a hurdle in the learning process. After some iterations, the agent manages to forget the pretrained weights and learn the new weights. The pretrained weights are supposed to be the reason of delayed convergence in the PnP approach. But the overall performance of both agents doesn't show much difference. But on the other hand, the No Depth Channel variant represented through the magenta line tells completely opposite story. The unavailability of the depth data (D) to this variant cause a serious negative impact on the success rate. The results show a drop of around 34% in the success rate as compared to the PnP approach. This deterioration in the performance of the No Depth Channel variant as compared to the PnP approach clearly proves the significance and importance of the depth data in the designed framework.

For the testing purposes, we design testcases on the basis of the number of objects in the clutter. We tested the performance of our proposed approach on three testcases having maximum 15 objects, 20 objects and 25 objects respectively. The best average success rate of around 85% was seen in the first test case where number of objects was kept less than 15. But as the number of objects increased in other two testcases the performance dropped a little. The average success rate of the second and third testcase were found to be around 81% and 78%. The major reason behind the performance drop in the testcases with more objects was the scarcity of grasping opportunities due to clutter becoming thicker as more objects were adjusted within the workspace. Even the clutter

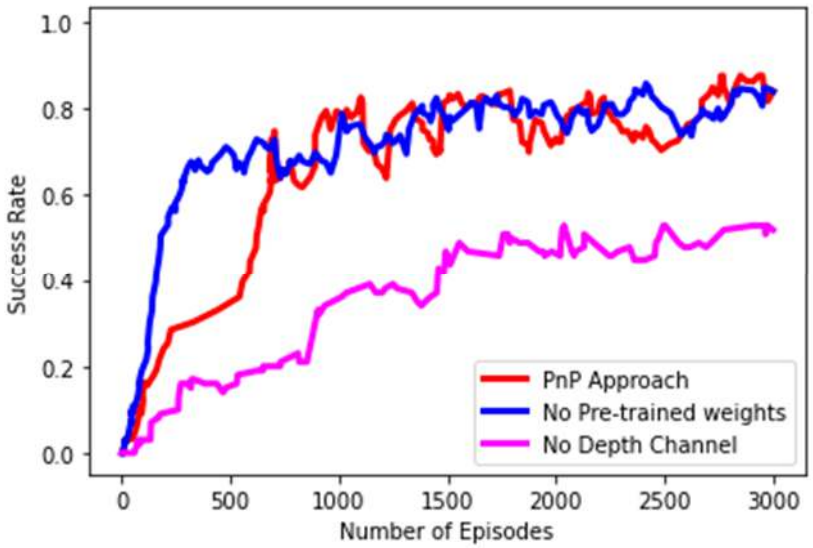

Figure 14 Performance comparison of our PnP approach with No Pre-trained weights and No Depth Channel approach

was ensured to be lightly packed still some objects get blocked by the neighboring objects causing problems in the grasping.

5.5.1 Discussion

We conducted different experimental trials in order to evaluate the performance of our self-learning pick-and-place framework. We compared the training performance of our PnP approach to the baseline approach the ResNet-based approach. Our PnP approach clearly performed better than the ResNet-based approach. The major suspected reason behind the better performance of the PnP approach as compared to the ResNet-based approach revolves around the feature learning mechanism of these two. The PnP approach utilizes the DenseNet-121 architecture whereas the ResNet-based approach deploys the ResNet-101 architecture as the Q-function approximator. In the ResNet architecture feature summation is performed but in the DenseNet architecture the features are concatenated instead of summation. This difference of feature summation and concatenation allows the PnP approach to gain better performance owing to the feature reusability characteristic earned because of feature concatenation in the DenseNet

architecture. The *L(L+1)/2* links between the *L* layers of the DenseNet play very vital role in the feature reusability. We also compared the performance of the PnP approach to the variants with no pretrained weights and no depth channel. The results show that there is not much role of the pretrained weights from the ImageNet dataset, in fact it negatively impacts by keeping the agent stuck in the local optima initially but as the new weights take over the agents starts learning but obviously experiences a delayed convergence as compared to the agent trained from scratch. The pretrained weights were not useful enough but the depth data was proven quite worthy when the PnP approach was compared with the No Depth Channel variant. The performance of the later one took a serious hit because of no height-from-bottom information. The lack of the depth information means that that agent has not intel to learn and differentiate the object in the foreground from its background which is critical for the grasping. The lack of depth data results into flaws in learning and inaccurate grasping moves, thus deteriorating the performance. In testing, it is found that the loosely packed clutters allow the agents to use their learning well and perform the pick-and-place at higher success rate. But as the number of objects keep increasing the workspaces becomes overpopulated and the clutter becomes thicker thus the grasping opportunities getting minimum. The objects get blocked by the neighboring objects due to the rush and the grasping opportunities vanish, which leads to the deterioration of the results. This limitation of the approach will be addressed in the Chapter 6.

5.6 Summary

In this chapter we proposed a self-learning RL-based framework for the industrial pick-and-place of regular and irregular-shaped objects in lightly packed clutter with the help of vision sensors. This approach presented in this chapter doesn't require object-specific geometric knowledge or domain-based knowledge beforehand. In this framework we designed, the agent learns to pick-and-place the regular and

irregular-shaped objects with the help of deep Q-networks and pixelwise parameterization technique. The Q-Function approximator, the end-to-end FCN, utilizes the pixelwise parameterization technique to predict the optimal 3D locations of the workspace for the robotic grasping. Rewards are awarded accordingly. Our results show that this proposed self-supervised RL-based framework enables the agent to gain learning and convergence to optimal policy within bounds of reasonable resources such as time. We also probe into the role and importance of various factors in the proposed approach such as the pretrained weights and the depth data from the vision sensors.

The main limitation of this approach is that it works well for only lightly packed clutters. The lightly packed clutters are those where neighboring objects are not too close to each other, thus not blocking the grasping opportunities for the robotic arm. Upon breaching this condition by increasing the number of objects and overpopulating the workspace area, testing results show deterioration of performance as grasping opportunities get blocked by neighboring objects. In order to address this limitation, we further extend this framework by involving more robotic manipulations so that agent can learn to pick-and-place without worrying about the clutter size and thickness in the Chapter 6.

Chapter 6

6 DRL-based Pick-and-Place in Clutter through Prehensile and Non-Prehensile Robotic Manipulation

6.1 Introduction

This chapter presents a DRL-based learning framework for robotic pick-and-place of regular and irregular-shaped objects using both prehensile and non-prehensile robotics manipulations with the help of vision sensors in an industrial environment. This chapter addresses our objective of developing DRL-based algorithms for learning and performing the vision-based pick-and-place tasks by combination of different robotic manipulations. The foundation laid in the previous chapter provides the basis of the vision-based multiple robotic manipulation presented in this chapter. We extend and enhance the framework presented in Chapter 5 in order to address the limitation of requiring only loosely packed clutters.

It has been evident that with the development in machine vision field the research focus has been mostly on vision-based approaches in terms of robotic manipulations. The robotic manipulations can be mainly divided into two types, the prehensile [120] and non-prehensile [105] robotic manipulation. The visualization of prehensile and non-prehensile manipulation from [121] can be seen in the Figure 15. An agent learning coordination between these both robotic manipulations can yield high success rate in the pick-and-place operations. The reason behind being that these both robotic manipulations can aid each other. The non-prehensile robotic manipulation can create grasping opportunities by producing gaps between the tightly-packed objects and on the other hand prehensile

manipulation enables more efficient, accurate and collision-free pushing operations. In Chapter 3, where we reviewed the relevant vision-based pick-and-place approaches in the existing literature, we also reviewed some approaches based on the principle of combining the prehensile and non-prehensile robotic manipulations for the robotic grasping or the pick-and-place tasks. This area of research shows that it can be fruitful in terms of results but still there is room for further exploration and improvement. Some of these approaches are perform push-grasping [105] i.e., pushing while grasping whereas other focus on pushing operations which bring the objects to predetermined grasping locations for pre-designed handcrafted grasping operations. The major downside to these approaches is that they are analytical approaches requiring object and domain-specific knowledge beforehand. There are a few model-driven approaches too but to consider the objects individually in the clutter they perform segmentation first, causing additional overhead and they also require some prediction system for the non-prehensile manipulations. Some approaches not considering the objects in the clutter individually, instead consider the workspace as whole suffer from bottleneck situation due to memory constraints as the network grows therefore can't handle multiple robotic manipulations efficiently. The approach proposed in

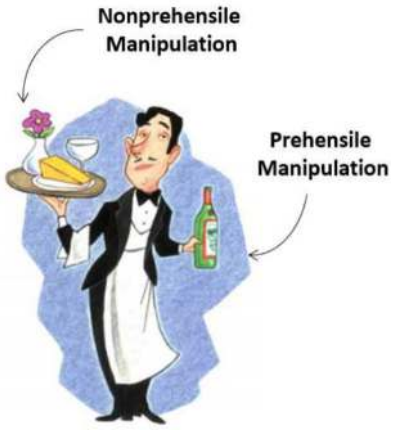

Figure 15 An example of Prehensile and Non-prehensile manipulations

this chapter is designed to address these research gaps in the existing relevant literature and the limitations of the existing relevant approaches i.e.

- Inefficient memory management
- Unidirectional pushing
- Additional overhead due to additional operations

In this chapter, we present a data-driven DRL-based self-learning framework for industrial pick-and-place of regular and irregular-shaped objects in a cluttered environment using both prehensile and non-prehensile robotics manipulations with the help of vision sensors. The aim of this approach is to make the RL agent learn and perform the pick-and-place of regular and irregular-shaped objects in order to increase the level of efficiency and throughput in various industries. Initially the MDP is designed as required in a RL problem, and then the model-free off-policy TD algorithm Q-learning is applied to it. We use pixelwise-parameterization method, where Q-values are calculated for each pixel, as discussed in Chapter 5. But unlike the previous approach in Chapter 5, we design separate Q-function approximators for each robotic manipulation involved. Each Q-function approximators consist of an extended memory efficient variant of DenseNet-121 architecture unlike our previous approach in Chapter 5 which uses the basic variants and remains vulnerable to bottleneck situations. **This technique of pixelwise parameterization has been used in a few other approaches** [79], [94] **too but these suffer from problems of bottleneck situation at the GPU end due to improper memory management for the quadratic feature growth.** As the state-space is large enough, we deploy an extended FCN based on DenseNet-121 architecture in an end-to-end manner for the Q-function approximation, so that the Q-values are calculated through this FCN. Rewards are awarded according to the success status of the agent. This proposed approach doesn't need any kind of geometric object-specific knowledge or any domain-specific information. This

approach also doesn't require any kind of extraordinary resources such as any large annotated datasets, any manual handcrafted information, etc. Unlike other mostly existing relevant approaches, the proposed approach considers the whole workspace as one instead of considering each object individually. Therefore, it doesn't require any additional steps causing any additional overhead such as segmentation or singulation for the identification of individual objects in the clutter. The promising training and testing results of this approach and its comparison with its some variants are also presented in this chapter. The key contributions of our extended framework can be listed as follows

- **A memory efficient data-driven DRL-based framework for the pick-and-place of regular and irregular objects** through combination of prehensile and non-prehensile robotic manipulations in sequential manner with the help of pixelwise parametrization scheme. This pixelwise parametrization scheme has been seen in a few recent studies but suffer from bottleneck situations at the GPU end due to quadratic feature growth and the networks lack memory management. This limitation is addressed in the proposed approach through the deployment of memory-efficient variants of deep neural networks.
- **Framework enabled bidirectional (left and right) non-prehensile robotic manipulations** whereas existing approaches are limited to unidirectional pushing mostly hard-coded and based on domain knowledge.
- **Minimizing the workload by skipping the additional operations causing additional overhead** such as segmentation, singulation, etc., required by other approaches in order to identify the objects individually in the clutter. The proposed approach considers the

whole workspace as one therefore doesn't require any additional steps, hand-designed features or heuristics.

6.2 Problem Scenario

The approach presented in this chapter considers the problem of pick-and-place with the use of vision sensors in a smart production line through sequence of prehensile and non-prehensile robotic manipulations. In this smart production line, a number of regular and irregular-shaped objects are being produced in a clutter and moved on a running conveyor belt. In this clutter - unlike the previous scenario in Chapter 5 of only loosely packed clutter - we allow the completely random allocation of the clutter in terms of object positions and orientations. The clutter can become as much tightly packed as possible randomly. We keep no checks regarding the gaps between the neighboring objects. At one end of the conveyor belt, where the robotic arm is installed for the pick-and-place operation, is our workspace. The workspace is monitored by the installed vision-sensors from the top angle. Once the random loosely packed clutter of regular and irregular-shaped objects reaches the workspace, and enters the field of view of the vision sensors, the conveyor belts stop for the pick-and-place operation. The vision sensors capture the scene of the workspace and feed it to our designed framework for the processing which will be explained in detail in the next sections of this chapter. Our framework outputs the optimal location for grasping or sliding in the workplace to the robotic arm. As soon as the robotic arm, receives the signal, it goes for the grasping or sliding, whatever action type is chosen. If the chosen action is grasping and it performs grasping successfully then it goes for the placement at the designated location, which is a basket in this case. But if the selected action is from non-prehensile manipulations, then the robotic arm performs the sliding. Rewards are awarded accordingly and the cycle continues. In this way the robotic arm, clears the whole clutter and then the

next clutter arrives for pick-and-place operation. In this way, this loop of pick-and-place operation continues.

6.3 MDP Design

As we described earlier, we propose to resolve the problem of learning the pick-and-place of regular and irregular-shaped objects using prehensile and non-prehensile robotic manipulations through RL. We create a joint framework of end-to-end deep neural networks through which our RL agents learn and converge to optimal policies. These multiple separate end-to-end deep neural networks are trained together simultaneously. In an end-to-end network, all steps starting from the initial layer to the final output layer are learnt by the model, thus making the memory management a crucial requirement in order to avoid the bottleneck situation.

Before resolving the task, we need to design the Markov Decision Process (MDP) of the task at hand. Generally, in any MDP, initially the agent can be at a state s_t at any timestep t. As the agent follows a policy π, the agent can transition to the next state s_{t+1} after taking an action a_t, meanwhile earning the reward $r_{at}(s_t, s_{t+1})$ and the cycle continues. The main goal behind this whole continuous process is to discover and learn such an optimal policy π^* which can ensure that maximum cumulative reward is scored in the future. The MDP designed for this approach which will be explained later in this section, is resolved through the model-free off-policy TD algorithm Q-learning. The Q-learning behaves as a greedy algorithm because it goes for highest action-value for the selection of action to be performed. An action-value can be defined as a measure of expected reward to be earned in the future if a certain action is taken in a certain state. These action-value are also known as Q-values and are calculated through the Q-function. A Q-function

can be depicted as $Q(s_t, a_t)$ where agent is currently at state s_t and chooses the action a_t to be performed at any timestep t.

Our designed MDP for this approach consists of the following major elements:

- **States**: Similar to the approach presented in chapter 5, our states in this MDP are actually RGB-D heightmaps. Each RGB-D heightmap is constructed from the current view of the workspace region on the conveyor belt. The current view of the workspace is received through the installed vision sensors. We deploy two types of the vision sensors known as orthographic visions sensor and perspective vision sensor. These both vision sensors are different from each other only because of their field of view. The orthographic vision sensors have rectangular field of view and the perspective vision sensor have trapezoidal field of view. The location of installed sensors and their field of views can be seen in the Figure 16. Both of these vision sensors gather RGB-D images of the workspace and then these images are merged together to increase the RGB and depth details. An RGB-D images consists of two pars, the RGB part which stores the color-based information through three channels red, green, blue (RGB), and the D part stores the depth information through only one channel, which we call as height-from-bottom channel (D). For the depth part of the RGB-D image each pixel is presented in relation to distance between the object in foreground and the background [116]. Once the RGB-D image showing the current state of the workspace is ready, respective RGB-D heightmaps are generated. RGB-D heightmaps are generated by following the technique presented in [94], where firstly the RGB-D data is projected onto a 3D point cloud and then it is back-projected upwards orthographically in the gravity direction. In this

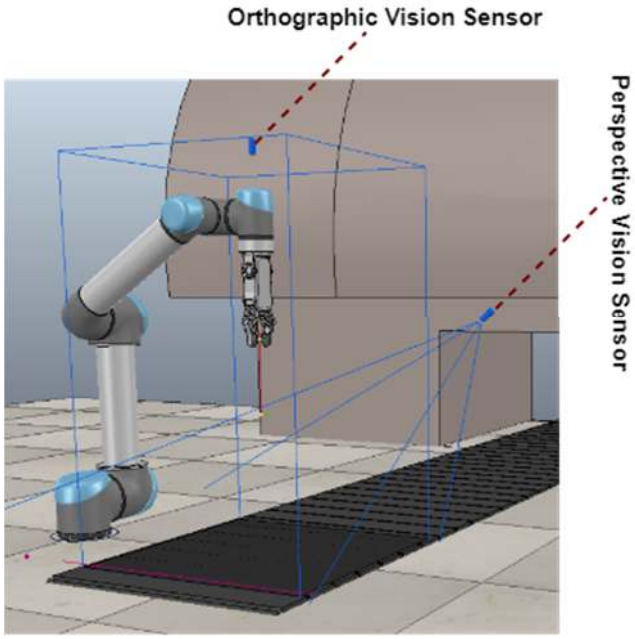

Figure 16 Vision sensors installation and field of view

manner RGB-D heightmaps are generated using both the color (RGB) and depth (D) information. This each RGB-D heightmap is actually our state s_t at any timestep t as shown in Figure 17. The RGB-D images used for the heightmap generation were of the workspace which is 224x224 pixel size, therefore the heightmaps or states are also of same size. Each state is actually composed of 50,176 individual pixels, where each pixel represents a unique 3D location of the workspace.

- **Actions**: This approach is designed for the pick-and-pace of regular and irregular-shaped objects through prehensile and non-prehensile robotic manipulations, with the motive behind that non-prehensile manipulations will increase the chances of success of prehensile manipulations. Therefore, we design three main actions Grasp, Left-Slide and Right-Slide. As evident from the names, the Grasp action is the prehensile robotic manipulation whereas the Left-Slide and Right-Slide actions

belong to the non-prehensile robotic manipulations category. For any of these actions a_t, in state s_t at any timestep t, which is a heightmap, a particular pixel p will be selected which have a corresponding 3D location p' in the workspace as described in Equation 1.

$$p \rightarrow p' \in s_t \qquad (1)$$

This p' is the location where the selected action will be performed by the robotic arm. In the case of the Grasp action p' is considered as the mid-point for grasping by the parallel-jaw gripper of the robotic arm from the top as our vision sensors are also top-mounted. The agent automatically adds 3 cm to the Z-axis component of the 3D location p', and then perform the grasping. Upon successful grasping, the grasped object is placed at the designated location i.e., the bin. But for the other both actions Left-Slide and Right-Slide, p' is considered as the initiation point of an 8cm linear push in the left or right direction, depending upon the action selection, using the tip of the robotic arm parallel-jaw gripper. Both types of action, grasping and sliding, can be seen in the Figure 18. Motion planning and path calculation for these actions are done on the

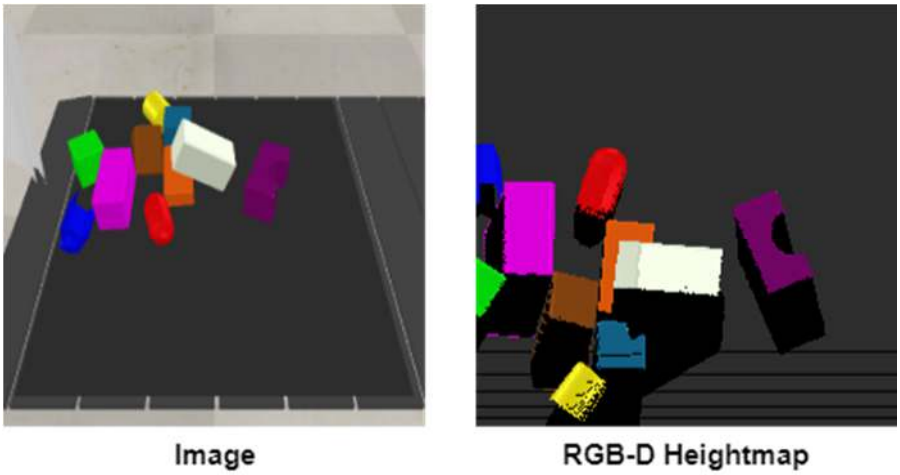

Figure 17 A sample of 224x224 RGB-D heightmap generated from the image of the workspace

runtime and saved for any future reuse. The motion planning module will be seen in a further section.

- **Rewards:** Rewards are awarded according to the success status of the action performed by the agent. In this scenario, if grasping is performed and object is placed into the bin successfully, the agent is rewarded is 1. If the grasping is successful, which is signaled by the infrared sensor between the jaws of the gripper, but placement fails then the agent is rewarded 0.8. If any of the sliding action is performed successfully, the agent is awarded 0.5. The success of the sliding action is measured through detecting any changes in the current workspace scene. If there is any changing in the scene, the sliding action is considered successful otherwise failure. In the case of failure in grasping or sliding the agent is rewarded 0.
- **Transition Function:** The transition function is responsible for keeping the transition records from one state to another. Our designed approach

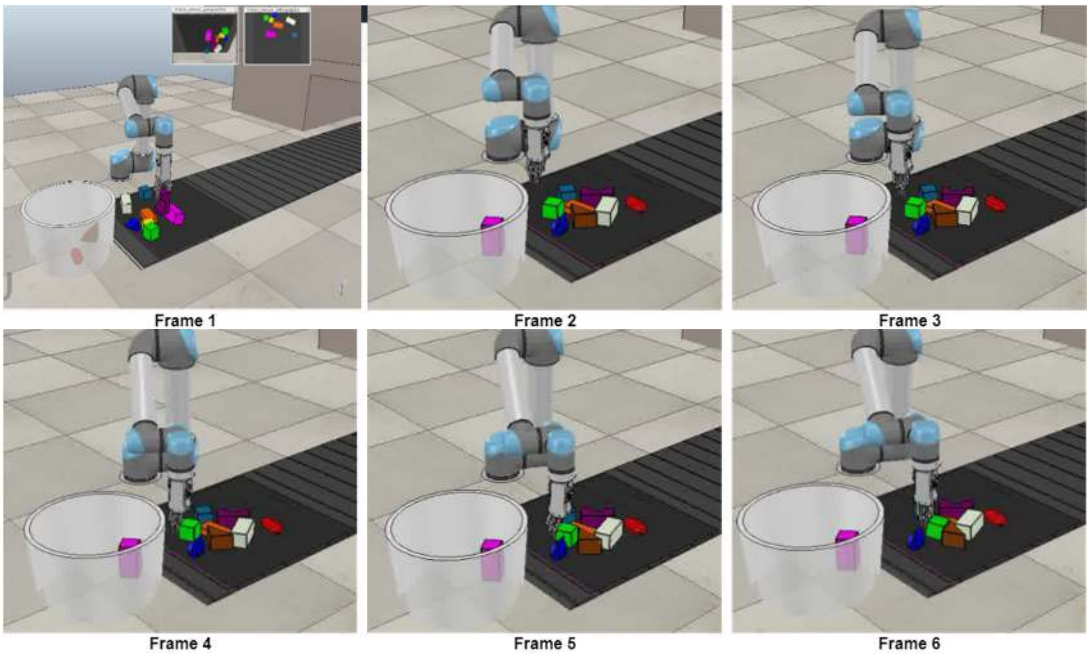

Figure 18 Grasping and sliding actions: Frame 1 (Grasping) Frame 2-6 (Sliding)

is completely model-free therefore it has no predetermined transition table, instead it is learnt through the experience. As in Chapter 5, we deploy the off-policy Q-learning TD algorithm, the Q-function is to be approximated to gain the Q-values. The Q-values, also known as action-values are actually the prediction of expected rewards in the future. But unlike Chapter 5, instead of using a single FCN as a Q-function approximator, we develop a joint framework of end-to-end deep neural networks. As we have three defined actions Grasp, Left-Slide and Right-Slide, we allocate each action a distinct FCN for the approximation of Q-values. This leads to a joint framework of three deep neural networks i.e. $\{\text{FCN}_{\text{Grasp}}, \text{FCN}_{\text{Left-Slide}}, \text{FCN}_{\text{Right-Slide}}\}$ which are trained simultaneously. For the Q-function approximation all three FCNs are supplied with the state(heightmap). Each FCN processes the state and outputs a pixelwise map of the Q-values. As the state is an RGB-D heightmap of size 224x224 pixels, the pixelwise map generated by each FCN as output is also of the same size. This means that each pixel of the state has a corresponding Q-value in output pixelwise map. In other words, the corresponding Q-value from the pixelwise map predicts the expected future reward if the corresponding action is performed at that pixel *p*'s corresponding 3D location *p'* in the workspace. These FCNs utilize the architecture known as DenseNet-121[117] as shown in Figure 19. We worked with its both variants, the pretrained and not trained. The factor which differentiates the DenseNet from other traditional architectures of FCNs is the nature of links between its layers. Usually, a general FCN having *L* layers have *L* links but the DenseNet with *L* layers have *L(L+1)/2* links among the layers. The *L* links mean that each layer is connected to its predecessor and successor but on the other hand the

L(L+1)/2 links mean that each layer is receiving feature-maps from its all predecessor layers and supplying its own feature-map to all its successor layers. This enhanced connectivity among the layers in the DenseNet tackles multiple problems like vanishing gradient, achieves parameter reduction and addresses other issues relevant to feature reuse and propagation. But the enhanced linking among the layers and a joint framework of three FCNs, can easily lead to quadratic growth of features risking the bottleneck situation as the networks grows deeper. The remedy to this memory concern deployed will be discussed in the later section. For each FCN we deploy the basic architecture of Dense-121 and extend it by adding two 1x1 layers along with ReLU activation and batch normalization. The whole multi-model designed approach can be seen in the Figure 20. Two trunks are used, where one trunk is supplied with the color data through color channels (RGB) and the other trunk is supplied with the depth data through depth channel cloned thrice to equate the number of color channels (DDD). The reason behind this cloning is to enable the agent use the pretrained weights learnt on the RGB images in the ImageNet dataset which are missing the depth channels. Features being learnt from both these trunks are concatenated and then further fed to the 1x1 convolutional layers. Moreover, a rotation function is also applied to RGB-D heightmaps before feeding to the FCN so that the degree of learning can be increased and multiple orientations can help in

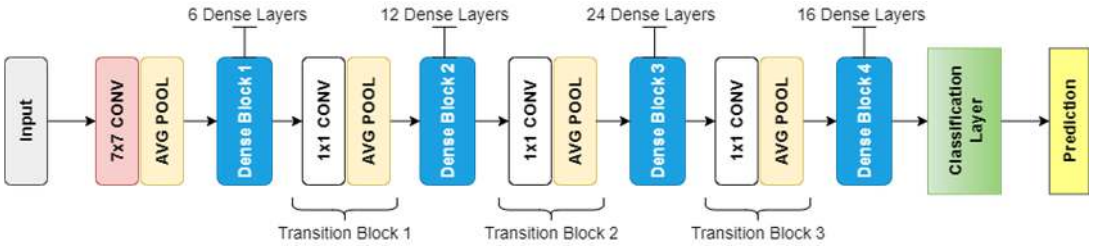

Figure 19 DenseNet-121 architecture

the process. While exploring the role of the rotation function, we first rotated the input state heightmap by 90 degrees and generated 4 rotated heightmaps. Later on, we rotated the input state heightmap by 45 and 22.5 degrees which led to generation of 8 and 16 rotated heightmaps as shown in the Figure 21. The first row in the Figure 21 presents the rotations at 90 degrees, appending second row to it constitutes the rotations at 45 degrees and all the four rows together represent the rotations at 22.5 degrees. The maximum feature learning ability while exploiting maximum possible orientations was experienced when the input state heightmaps were rotated at 22.5 degrees generating 16 rotated heightmaps. With this setting of rotation function, each FCN gets supplied with 16 rotated heightmaps and generate 16 pixelwise maps as outputs. So, for each pass total 48 pixelwise maps are generated, out of which 16 maps are for the Grasp action, 16 maps are for the Left-Slide action and 16 maps are for the Right-Slide action. For the action selection, out of the total 48 pixelwise maps,

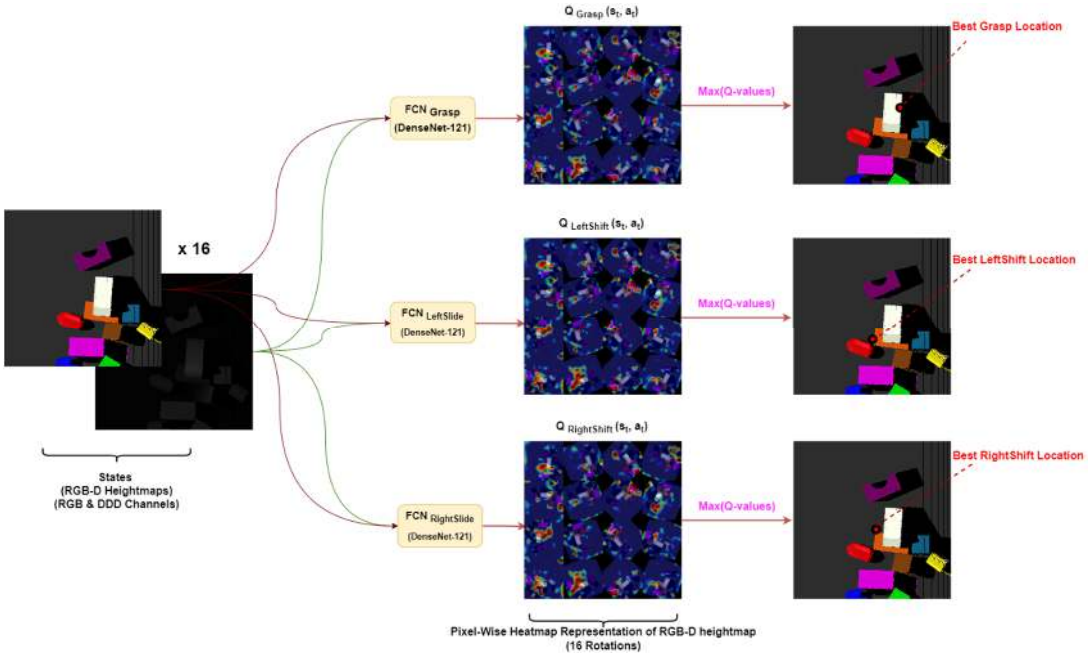

Figure 20 The flow of prehensile and non-prehensile robotic manipulation-based approach

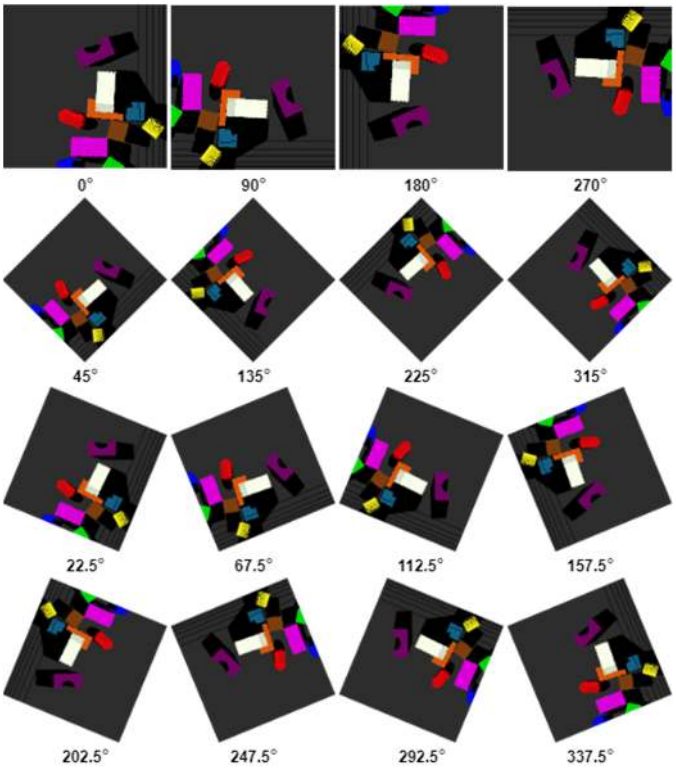

Figure 21 Rotation function for RGB-D heightmaps (first row:90 degrees, first and second row: 45 degrees, all rows: 22.5 degrees)

the map having the maximum Q-value, action corresponding its FCN is chosen. The action is performed at the 3D location *p'* corresponding to the pixel *p* having the maximum corresponding Q-value in the pixelwise map as shown in Equation 2.

$$max\ Q(s_t, a_t) = max\left(FCN_{Grasp}(s_t),\ FCN_{Left-Slide}(s_t), FCN_{Right-Slide}(s_t)\right) \quad (2)$$

These FCNs are considered quite efficient Q-function approximators in the case of pixelwise situations owing to efficient computation schemes. In each forward pass, all three FCNs collectively calculate around 2,408,448 Q-values within reasonable duration. The total number of Q-values in each forward pass are calculated as 48 pixelwise maps of size 224 x 224, which in total amounts to be 224 x 224 x 48. With the help of this pixelwise parameterization scheme along with the designed rotation

scheme helps the agent to learn and converge with reasonable amount of training data.

- **Discount Factor:** The discount factor and is kept between 0 and 1 in order to increase the importance of the future rewards and make the agents learn at faster pace.

6.4 Training Specifications

In this proposed approach we train our RL agents to learn pick-and-place of regular and irregular objects through prehensile and non-prehensile robotic manipulations with the help of the vision sensors.

In each iteration, the vision sensors capture and merge the RGB-D images of the current state of the workspace. Then from this image the RGB-D heightmap is generated. This heightmap is then fed to the rotation function which was discussed in the previous section. As the output, the rotation function generates rotated heightmaps. These rotated heightmaps are supplied to each FCN as input and then each FCN generates pixelwise maps of Q-values as the output. Each pixelwise map consist of around 50,176 values, therefore due large volume of values these pixelwise maps can be represented through the heatmaps. The Figure 22 shows the pixelwise maps of the 16 rotated heightmaps (22.5 degrees) in the heatmap format. The areas appearing hotter are actually the pixels with the higher Q-values.

In the backpropagation phase we utilize the Huber Loss function [118], which proves to be a decent trade-off between the mean square error (MSE) and the mean absolute error (MAE). In the backpropagation, we only pass the gradient for the pixel p in the particular FCN which generated the highest corresponding Q-value in the pixelwise map. For all the remaining pixels zero loss is backpropagated. Unlike the approach propose in Chapter 5, FCNs are trained using stochastic gradient descent with momentum [122] variant, which is considered more efficient and rapid

as compared to traditional stochastic gradient descent (SGD), and the reason being that traditional SGD calculates noisy derivatives of loss but with momentum variant, exponentially weighted averages are involved which are more close to actual loss instead of SGD's noisy values, thus aiding gradient vectors move in right directions leading to faster convergence. According to the training specifications of our network values of momentum, learning rate, and weight decay were set at 0.9, $1e^{-2}$, and $1e^{-3}$ respectively. In our training phase, we also benefit from the feature of experience replay, where the agent keeps a record and reuses its experiences from the past. We follow the stochastic prioritization approach developed in [123] for the experience replay. In order to address the exploration vs exploitation dilemma, we deploy epsilon-greedy action selection, where the epsilon value is initialized at 0.3 and gradually decays down to 0.1. In the terms of hardware, this our designed framework has been trained on NVIDIA GeForce RTX 2080/PCIe/SSE2 with dedicated 8 GB memory using Intel® Core™ i79700K CPU @3.60GHz × 8 processor along with 16 GB RAM. For the development of the models and their training we used PyTorch 1.7.1 along with CUDA 11.4, cuDNN 8.2.2 and the NVIDIA Drivers 470.130.01.

DenseNet is considered as one of most effective and efficient neural network architecture due to its maximized linking [$L(L+1)/2$] and enhanced feature map sharing among all the layers leading to maximum feature reuse, However, despite being efficient, if proper memory management is not ensured it becomes vulnerable to bottleneck situation at the GPU end due to quadratic growth of its feature maps because of its contagious convolutional operations. Therefore, in order to address this vulnerability of the base variant, we deploy the a memory-efficient variant of DenseNet-121 as in [124] where for the operations such as gradients, batch normalization and concatenation memory sharing schemes are utilized. In the deployed memory-efficient DenseNet-121 network the outputs generated from the

concatenation, batch normalization and the ReLU layers are stored in the temporary storage buffers instead of doing new memory allocations as in the base variant. In this manner the risk of the bottleneck situation due to the quadratic growth of the features as the network depth increases is avoided.

When the simulated environment is initialized for the training, the workspace area designated on the conveyor belt contains a random clutter of N regular and irregular-shaped objects. Unlike the approach proposed in Chapter 5, there is no requirement for ensuring the clutter to be loosely packed instead the clutter is completely random and can be as much as tightly packed as possible. Once the clutter is ready, the agent starts the pick-and-place operation and keeps continuing until the workspace is cleared. When the whole clutter has been placed inside the bin, the designated placement location, it is considered as one complete iteration. Immediately the next iteration starts and the random clutter reaches the workspace for the pick-and-place and the loop continues. In our training phase, we set the N=10, meaning 10 random regular and irregular-shaped

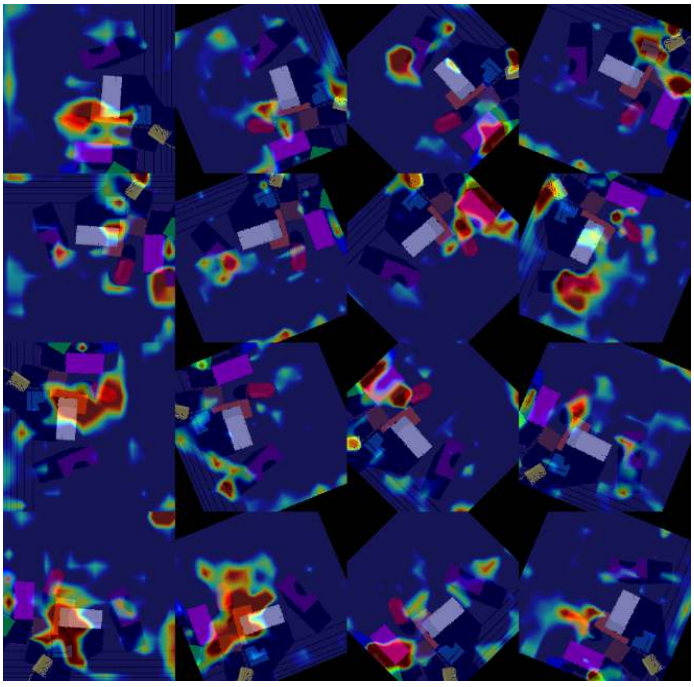

Figure 22 Heatmap representation of 16 rotations of RGB-D heightmap at 22.5 degrees

3D objects (as shown in Figure 23) randomly arranged in each clutter throughout the training cycle. The length of the training cycle is set at 3000 iterations.

6.4.1 Motion Planning Module

Today in the field of robotics, a number of motion planner control solutions are available. For our proposed approach, different motion planning libraires and frameworks were considered and reviewed such as Trajpot [111] and OpenRave [112]. Among all other considered option, Open Motion Planning Library (OMPL) [113] proved to be the most appropriate option due to its maximum degree of customization allowance. In OMPL, multiple control-based planners and geometric planners are available. Some of the popular sampling-based planners available in OMPL are Expansive Space Trees (EST), Single-query Bi-Direction Lazzy (SBL), Probabilistic Roadmap Method (PRM), Rapidly-exploring Random Trees (RRT), etc. The OMPL's planner deployed in our approach is a single-query planner. It is known as RRT-Connect [114], which happens to be a bi-directional variant of the Rapidly-exploring Random Tree (RRT) planner. The main idea of the RRT-Connect planner is to create two RRTs, where one is located at the start point and the other one is located at the end point, and then connect both. Utilizing two RRTs simultaneously is the reason behind its ability to outperform the basic RRT planner.

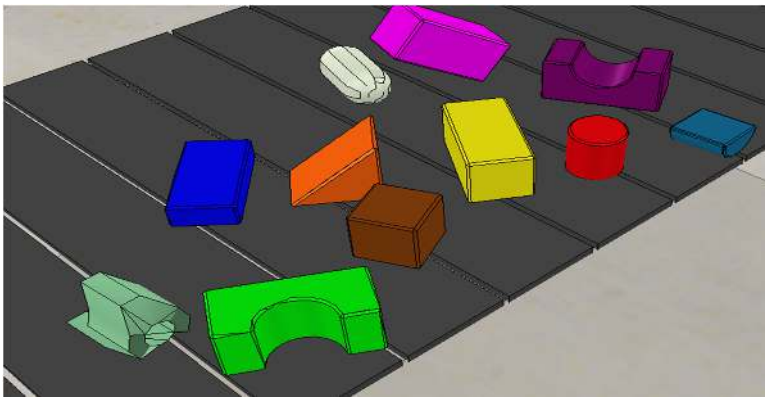

Figure 23 Ten regular and irregular-shaped randomly placed 3D objects

6.4.2 Simulation Environment

According to our task requirement and resources, the Virtual Robot Experimentation Platform (V-REP) proved to be the most suitable physics engine-based simulator. The V-REP is actually a 3D robotic simulator which involves the integrated support for development and coding [30]. It is equipped with multiple physics engines such as ODE and Bullet variants in order to achieve the real-time emulation of all the objects during the simulations. The availability of multiple APIs and the functionalities such as threaded/non-threaded Lua scripting enables the V-REP to develop cross-platform application through combining platforms such as Python, C++, Java, etc. With the help of the python-based API, it was possible for us to create a python-based RL agent which was able to interact with the environment inside the V-REP being controlled through the Lua scripts.

V-REP also provides the Forward and Inverse Kinematics calculation module. In Forward kinematics, joint parameters are supplied as input to the kinematics equations and the position of the end-effector is calculated as the output. On the contrary, in the Inverse kinematics, the required position of the end-effector is supplied as input and the joint parameters are calculated as the output [115]. Another importation module of V-REP utilized in this approach is the collision-detection and collision-avoidance module which detects and avoids any possible collision of the robotic arms with the environment. In our experimentation trials, we used the famous UR5 robotic arm of Universal Robotics which is a 6 degree-of-freedom (DOF) robotic arm. For the grasping part, we attached a RG2 gripper to our UR5 robotic arm.

6.5 Experimental Results & Discussion

In this section we conduct a series of experiments in order to evaluate the learning and performance of our self-learning framework in the pick-and-place task through

prehensile and non-prehensile robotic manipulations. The designed simulated environment for the training and testing can be seen in the Figure 24.

The approach we designed and described in previous section is named as G&S approach, the abbreviation for Grasp and Slide. In order to assess the performance of the proposed G&S approach we need a baseline approach to compare with. As our G&S approach is a RL-based approach, we develop a DL-based approach for the comparison similar to the DL-approach presented in [94] where they used a manually annotated dataset. The baseline approach also utilizes the pixelwise parameterization technique but acts as a self-supervised binary classification scheme. There is no RL element involved in the baseline approach, therefore no rewarding schemes. The concept of states and actions remains same as in the proposed G&S approach. Three FCNs corresponding to three actions being trained simultaneously. The main difference from the G&S approach is that the FCNs in this baseline approach will be trained to through binary classification (Successful and Unsuccessful) in self-supervised manner while generating pixelwise map of affordances. This baseline approach is considered as self-supervised approach because there is not ready annotated dataset instead labels are generated at the runtime according to the success status of the performed action. For instance, if the grasping action is performed at a certain pixel with highest affordance value in the pixelwise map and the object is picked and placed successfully in the bin, the 'successful' label will be generated. If the pick-and-place fails the 'unsuccessful' label will be generated. In the same manner, in the case of Left-Slide or Right-Slide action if change is detected in the scene of workspace, 'successful' label will be generated otherwise the label will be 'unsuccessful'. In the backpropagation phase, similar to the G&S approach, gradient passing is only performed in the chosen action's corresponding FCN's pixel in the pixelwise map with highest affordance value. For all the remaining pixels zero loss is backpropagated. This baseline approach follows a greedy policy as it goes for the action yielding highest

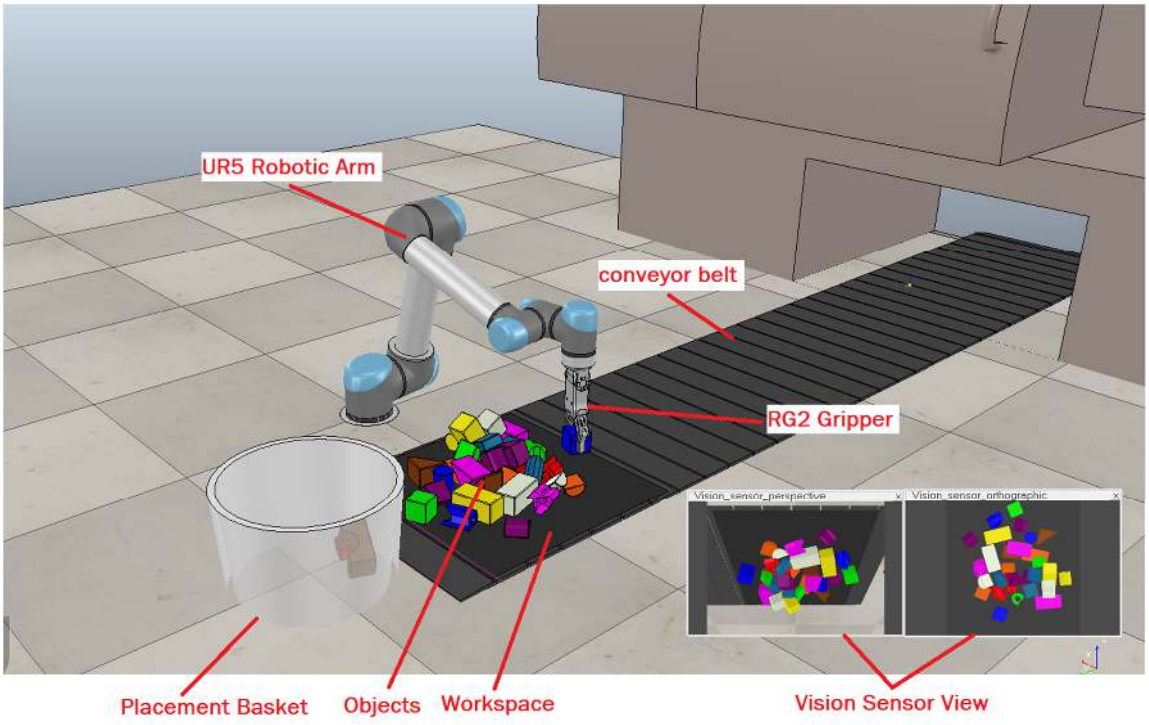

Figure 24 Simulation Environment designed for training and testing

affordance value in the pixelwise maps at any timestep. The performance comparison of our proposed RL-based G&S approach and the DL-based binary classification baseline approach can be seen in the Figure 25. In both approaches the agents experienced 3000 episodes in the training, where each episode consisted of a random clutter of 10 regular and irregular-shaped objects. An episode ends when the workspace is cleared of all the objects or the agent experiences 10 consecutive unsuccessful actions. A failed grasping or a slide yielding no change to the workspace scene is considered as unsuccessful action. The blue line presents the G&S approach with a success rate of around 84% whereas the green line shows the success rate of the binary classification baseline approach around 57%. The success rate is calculated as the number of objects placed in the bin successfully over the number of actions performed in the whole episode. A suspected cause behind low success rate of the baseline approach as compared to our G&S approach is that the baseline

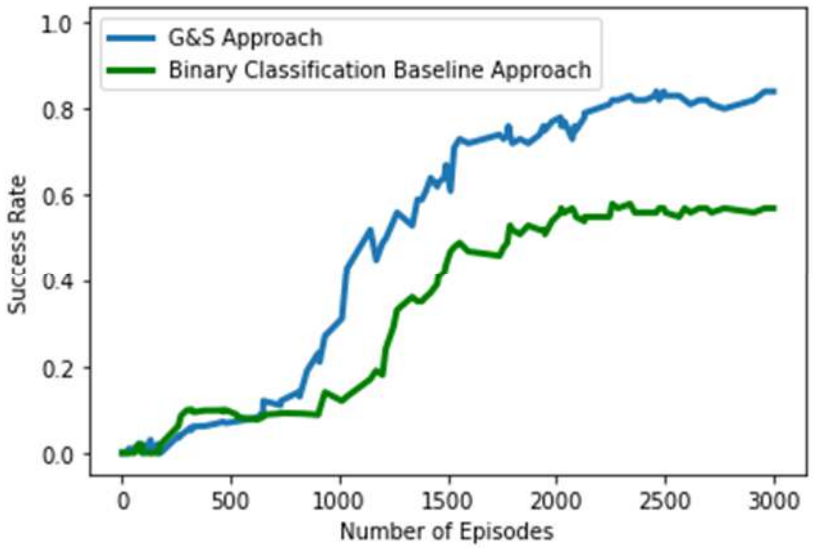

Figure 25 Performance comparison of G&S approach with Binary Classification Baseline approach

approach goes greedy without forming and following any strategies to learn the sequences of actions for future as our G&S approach do with the help of the reward schemes.

Aside from the DL-based approach, we also designed a RL-based approach similar to the proposed G&S approach. The difference being the change of neural network architecture. This variant is designed by utilizing the ResNet-101 architecture with pretrained weights on ImageNet [125] dataset instead of the DenseNet. A ResNet [126] consists of residual blocks. The ResNet is designed in order to resolve the issue of the vanishing gradient problem which occurs when the depth of the network exceeds the threshold value after continuous addition of layers to the network. To address the vanishing gradient problem the ResNet utilizes skip connections. The skip connections are actually the identity shortcut connections which provide better flow of gradient through skipping a few layers. The performance graph of the ResNet-101-based approach against the G&S approach is given in the Figure 26. The graph shows that the ResNet-101-based approach has been outperformed by our proposed G&S approach by a margin of around 13%. The possible explanation behind this lack of performance by the ResNet-101-based approach is the fact that ResNet implements summation of feature maps but on

the other hand, the DenseNet performs concatenation of the feature maps, Due to the ability to concatenate feature maps, DenseNet provides the maximum degree of feature reusability. The skip connections or the identity shortcut connections in the ResNet minimizes the representation strength but the DenseNet's concatenation through the $L(L+1)/2$ links maximizes it.

The basic aim behind combining the prehensile and non-prehensile robotic manipulations in this learning-based framework is to enable the agent to learn the sequence of both manipulations in order to facilitate the task of the pick-and-place and reach the convergence. In this framework we tried to explore the learning of non-prehensile robotic manipulations (Left-Slide and Right-Slide) in such a manner which can lead to the increase of rate of success of the prehensile robotic manipulation (Grasp) and vice versa. In order to evaluate the contribution of non-prehensile robotic manipulations (Left-Slide and Right-Slide) to the learning process of the pick-and-place task, we trained another variant of the proposed G&S approach. The variation between the G&S approach and this variant is that we pulled the rule of 0.5 reward for successful non-prehensile manipulation from the designed reward scheme. We ensured that no reward is awarded

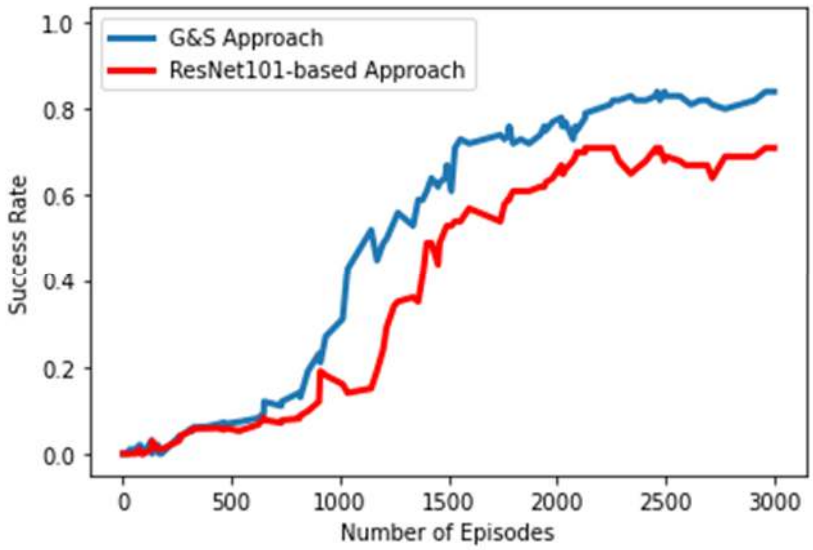

Figure 26 Performance comparison of G&S approach with ResNet-based approach

for the Left-Slide or the Right-Slide actions. Reward scheme for the Grasp actions remained untouched. The Figure 27 shows the performance of the No Sliding Rewards variant in the cyan line against the proposed G&S approach in the blue line. Making the non-prehensile robotic manipulation rewardless for the agent results in as drop in performance of around 22%. This drop in performance depicts the importance of the sliding operations for the grasping operations in our scenario. The graph shows delayed rise in the curve of performance which is evident for slow rate learning of the agent due to no rewards being provided the non-prehensile manipulations, therefore failing to understanding the importance of sliding operations in the pick-and-place task. We have already seen the performance deterioration when only grasping operations were available for the tightly packed clutters in the Chapter 5. With the absence of rewards for sliding operations, the agents learn a grasp-only kind of policy where the sliding importance can be perceived by the agent only indirectly when future grasping reward is earned after the rewardless sliding action. Therefore, the delayed curve rise as compared to the G&S approach represents the slow rate of improvement in the policy.

As we discussed in previous sections that we implemented the DenseNet-121 extended architecture in both manners, with pretrained weights and trained from scratch too. The G&S approach utilizes the pretrained DenseNet-121. The pretrained variant is trained on the subset [127] of the famous ImageNet dataset. This subset covers around 1000 various object classes and contains around 1,431,167 images for training, validation and testing. In order to understand and evaluate the role of these pretrained weights in the overall learning process and the convergence of the policies by the agent, we created another variant of our G&S approach. In this variant, the DenseNet-121 extended architecture network was trained from scratch and no pretrained weights were involved. The Figure 28 presents the performance comparison of the proposed G&S approach in the blue color which utilizes the pretrained variant and the no pretrained weight approach in the orange color. It is evident from this performance comparison that the pretrained

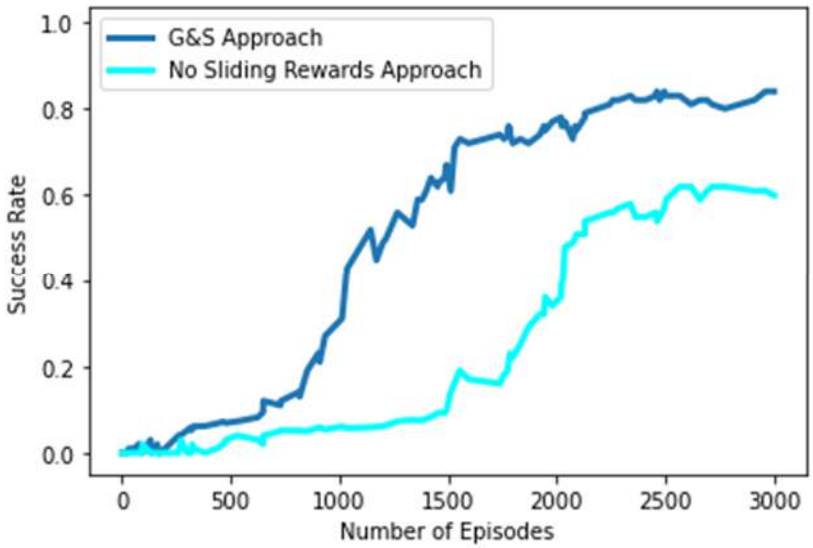

Figure 27 Performance comparison of G&S approach with No Sliding Rewards approach

weights from the ImageNet dataset doesn't play much role in the learning of the agent because both variants show almost similar progress. Whereas usually the reason behind involving pretrained weights in training a network is to achieve faster convergence and higher accuracy while consuming minimal training time. But in our case, we didn't get any sort of such boost from the pretrained weights. The probable cause for this in our opinion is the difference among the pixels pattern of our generated RGB-D heightmaps and the real-life images from the ImageNet dataset. The data we deal with is synthetic data whereas the pretrained weights were learnt after training on the real-life data. In fact, the early curve rise in the orange line as compared to blue line shows that the variant with no pretrained weights was not stuck in some local optima and started learning soon. Whereas the G&S approach was stuck with the pretrained weights which were causing hurdle instead of the expected boost, after some time gradually it grew out of those pretrained weights and started making progress.

Another element whose importance was explored and evaluated was the depth data in the RGB-D images from the vision sensors. The two trunks were designed in order to entertain both the depth information and the color information. The depth information

which was supplied through the depth channel (DDD) where the color information was provided by the three-color channels (RGB). In order to evaluate the role of the depth information, we implemented another variant of the G&S approach, which was deprived of the depth channels. This no depth channels variant was only supplied with the color (RGB) images and heightmaps. As no depth data was to be entertained in this variant, therefore only on trunk dealing with color data was enough in the FCNs of this variant's Q-function approximator. The Figure 28 presents the performance of the no depth channels variant in magenta color. Results show severe performance deterioration as the success rate falls to around 51%. The absence of the depth information means the lacking of the height from the bottom information which is crucial to differentiate the foreground and background in order to position the object. Once we deprive depth information, it means no features can't be learnt regarding the depth element. This drop in the performance clearly highlights the vital importance of the depth channels and height from the bottom information for the agent and tells how critical these are for accurate learning of the policies and their optimal convergence.

In order to evaluate the learning of the agent, we tested it under unseen circumstances. As we described earlier that when the simulated environment is initialized, the workspace area designated on the conveyor belt contains a random clutter of N regular and irregular-shaped objects. Once the clutter is ready, the agent starts the pick-and-place operation and keeps continuing until the workspace is cleared. When the whole clutter has been placed inside the bin, the designated placement location, it is considered as one complete iteration. Immediately the next iteration starts and the random clutter reaches the workspace for the pick-and-place and the loop continues. In our training phase, we set the N=10, meaning 10 random regular and irregular-shaped 3D objects randomly arranged in each clutter throughout the training cycle. The length of the training cycle is set at 3000 iterations.

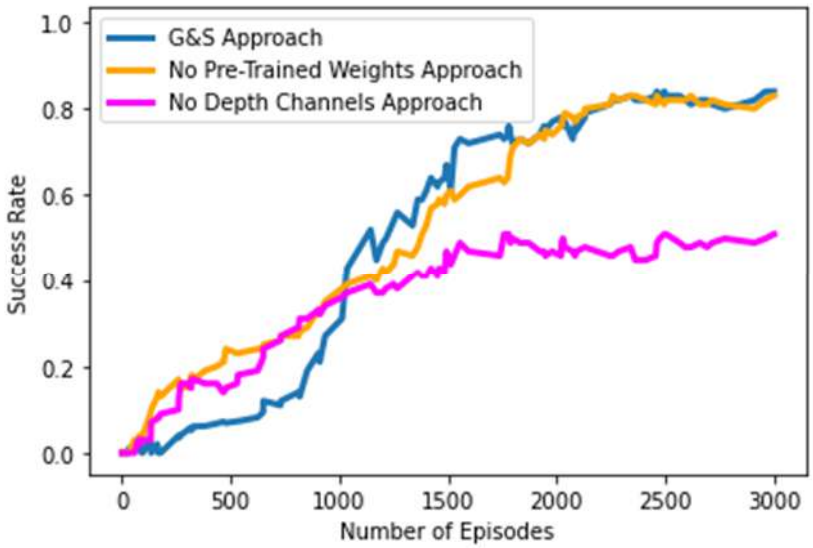

Figure 28 Performance comparison of G&S approach with No Pre-trained weights and No Depth Channels approaches

The G&S agent trained in the abovementioned manner was tested on clutters with varying densities. To categorize the varying densities, we develop three different categories of the clutters as shown in the Figure 29. We call the first category as minimum density clutter comprising of 6 to 10 random regular and irregular-shaped objects. In the second category we set the objects limits from 15 to 20 regular and irregular-shaped objects and name it as medium density clutter. The third and the last category is called the maximum density clutter and contains 25 to 30 random regular and irregular-shaped objects. Unlike the approach presented in chapter 5, these clutters are not ensured to be loosely packed, instead the thickness is kept completely random.

Along with this density-based clutter testing, we also design various complicated scenarios for testing the performance of the G&S trained agent. In these complicated scenarios the object count ranges from at least 3 to at most 7. In each of these complicated scenarios, a few objects may be easy to grasp but most of the objects are in some kind of locking manner as shown in Figure 30 which is tricky for the agent and create maximum resistance for the agent to pick-and-place the objects in the workspace. The average

performance success rate for the density-based clutter and complicated scenarios testing is shown in the Table IX.

Table IX Average Testing Results (%)

Category	Success Rate	Grasping Success
Minimum Clutter	84%	96%
Medium Clutter	82%	95%
Maximum Clutter	74%	82%
Complicated Scenarios	65%	73%

In the results above, the average grasping success rate is also mentioned for all individual categories. The grasping success rate of each category is calculated as the number of successful grasp actions over the total grasps attempted in the complete episode. The results show that our G&S agent performs well overall in the case of minimum and medium density clutters but the performance dropped in the overall success rate and grasping success rate both in the maximum density clutters. The major visible reason behind this drop in performance is the objects getting placed over other objects in weak poses due to random arrangement in maximum density. This can also be witnessed in the maximum density clutter shown in Figure 29. The trouble of these weak posed objects placed over other objects is that as the robotic gripper RG2 forms contact with them to

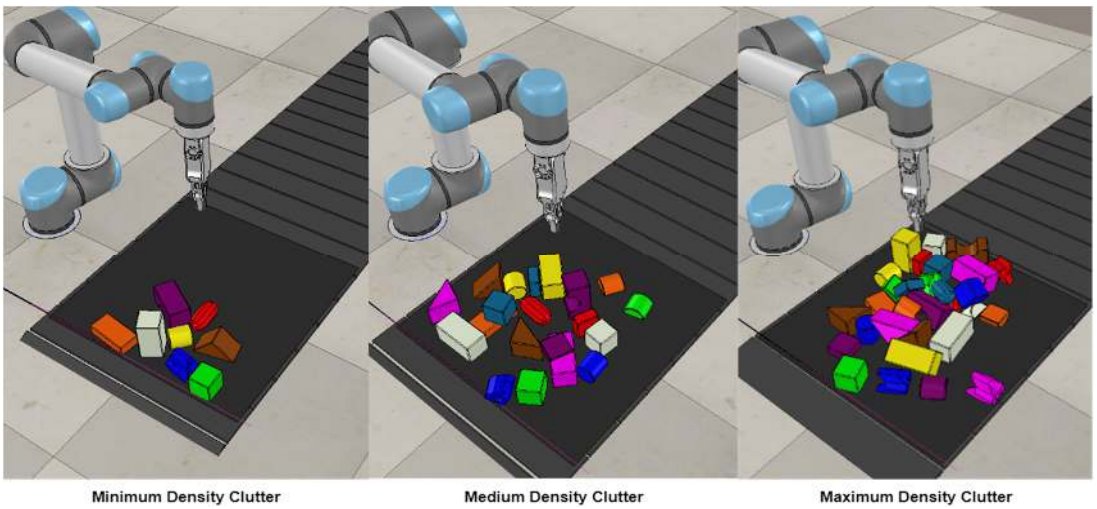

Figure 29 Categorization on the basis of clutter density

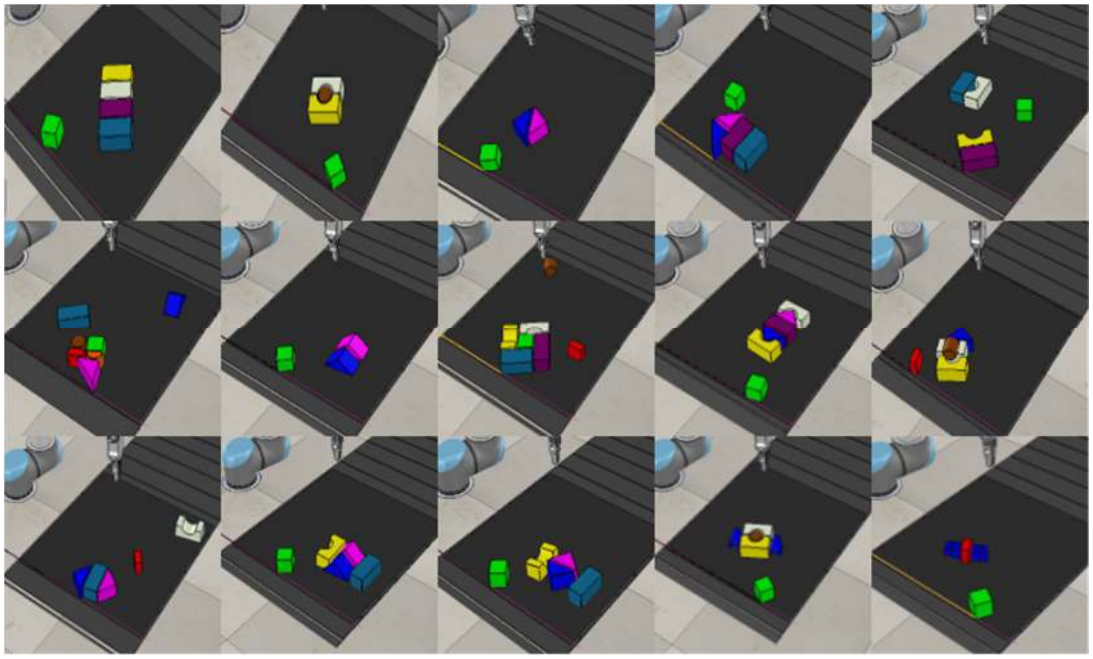

Figure 30 Complicated scenarios designed with 3-7 regular and irregular-shaped objects in locking manners

grasp, they slip due to weak balance and fell over thus failing the grasping attempt and lowering the success rate too. In the complicated scenarios, the objects are placed in such locking manners which can make the agent face maximum resistance while pick and placing the objects from the workspace. For instance, for some objects the gap to grasp becomes quite narrow whereas for some objects the gap may be too wide. These designed scenarios were completely new for the agent as the training was carried out on random clutters not such complex scenarios. Another major reason for lower success rate can be seen as the smaller number of objects in the workspace. For instance, if the workspace has only 3 objects in it and the agent successfully clears the workspace in only 4 actions. The agent has performed well in this scenario but according to the success rate metric it will be calculated as (3/4*100=75%). With respect to all this factors, we can say that our agent has performed quite well in our testing phase

6.5.1 Discussion

We conducted a series of simulated experimental trials to evaluate the performance of our RL-based self-learning agent for the pick-and-place of the regular and irregular-

shaped objects with the help of prehensile and non-prehensile robotic manipulations. The main goal behind these experimental trials was to evaluate the learning ability and convergence of the agent and the role of different factors in this learning. The overall results of our training and testing trials show that the agent successfully learns the sequence of prehensile (Grasp) and non-prehensile (Left-Slide and Right-Slide) robotic manipulations in order to pick-and-place the regular and irregular-shaped objects from the random clutters.

Our proposed agent, G&S agent clearly proves its ability to learn the sequences of prehensile and non-prehensile robotic manipulations when it is compared to the DL-based binary classification baseline approach. The suspected reason behind this lead of our G&S agent is its ability to learn the optimal sequence of prehensile and non-prehensile robotic manipulations with the help of the rewarding scheme. In other words, the agent successfully learns to draw the link between the immediate major reward earned over successful grasping after earning the minor award over sliding. The role of the DenseNet-121 architecture for the Q-function approximation in the learning and convergence of the proposed G&S agent can also be witnessed when it is evaluated against the ResNet-101-based variant. The results show that proposed G&S agent has clearly an edge over the opponent due to the feature reusability trait through the feature concatenation capability of the DenseNet architecture owing to the maximum linking $L(L+1)/2$ among the layers. On the other hand, ResNet lacks this edge due to its feature summation policy instead of the feature concatenation.

In order to verify the worth of the non-prehensile robotic manipulations in the learning and convergence of the agent, we tried killing the rewards for the non-prehensile manipulations which resultantly damaged the overall performance of the agent proving the worth and role of the non-prehensile manipulations. The proper rewarding for the non-prehensile robotic manipulations helps the agent to learn the optimal sequences of grasping and sliding operations for the pick-and-place of the objects. In addition to these

we also evaluated the role of the pretrained weights and the depth information gained from the depth channels of the vision sensors. The role of the later one was found very crucial and critical to the overall success of the operation, whereas the role of the prior one was of not much worth, in fact discarding the pretrained weights made the agent learn earlier. The absence of the depth data led to the failure of agent's ability to perceive the object under consideration separated from the background and foreground as the distance information was not available. This led to flawed learning and inaccurate grasps deteriorated the performance rate of the agent.

The results of the density-based and complicated scenario testing clearly show the agent's success in learning the sequences of prehensile and non-prehensile robotic manipulations and the convergence to the optimal policy. Despite the factors like overpopulated workspace, weak posed objects overlapping each other and complex locking pattern scenarios still the proposed G&S agent displays good performance and proves the worth of its learning and convergence during the training phase.

6.6 Summary

In this chapter we proposed a self-learning RL-based framework for the industrial pick-and-place of regular and irregular-shaped objects in clutter with the help of prehensile and non-prehensile robotic manipulations using the vision sensors. This approach presented in this chapter doesn't require object-specific geometric knowledge or domain-based knowledge beforehand. Unlike the approach presented in Chapter 5, this approach is not limited to the lightly packed clutters only, instead it can be subjected to any sort of random clutter.

In this framework we designed, the agent learns to pick-and-place the regular and irregular-shaped objects with the help of sequence of prehensile (Grasp) and non-prehensile (Left-Slide and Right-Slide) robotic manipulations using deep Q-networks and

pixelwise parameterization technique. For Q-function approximation, joint training framework is designed consisting of three individual end-to-end FCNs comprising of memory-efficient variant of DenseNet-121 architecture. Each FCN corresponds to each action and predicts the optimal pixel and its relevant 3D location of the workspace for the particular action. Rewards are awarded accordingly. Our results show that this proposed self-supervised RL-based framework enables the agent to gain learning of sequence of prehensile and non-prehensile robotic manipulations and the convergence to optimal policy. This approach deploys a memory-efficient neural network architecture therefore is not vulnerable to bottleneck situation like other approaches in the past due to lack of proper memory management. Unlike other approaches, it also enables agent to learn bi-directional (left and right) non-prehensile manipulation and doesn't incur any additional overhead for any additional computation such as segmentation, singulation, etc. We also probe into the role and importance of various factors in the proposed approach such as the non-prehensile manipulation rewards, pretrained weights and the depth data from the vision sensors.

Chapter 7

7 Conclusion and Future Work

The robotic arms used in industry today are hard-coded and only follow the set of instructions fed to them in order to perform the task. In case of any modification or a new task, the robotic arm needs new code. This shortcoming has been the focus of the research community from last few years. This thesis also addresses the same research gap by proposing a learning-based data-driven framework for the industrial robotic manipulations such as grasping and pick-and-place. This thesis considers a wide range of robotics-based industrial setups where some deploy the visions sensors and some use non-visual sensors such as proximity sensors. RL-based agents are trained using the off-policy i.e., Q-learning and on-policy i.e., SARSA TD algorithms for learning the industrial robotic pick-and-place. Resultantly, the agents learned to pick-and-place regular and irregular-shaped objects in clutter are at the designated location with the help of prehensile and non-prehensile robotic manipulations.

7.1 Contribution

The core contribution of this work is to propose, intelligent, self-learning pick-and-place frameworks that can lead to enhanced results through increased efficiency and throughput. In order to achieve this target, we identified three main challenges to complete. The first challenge was to develop a learning-based framework for learning the industrial pick-and-place for the setups where there

were no vision sensors. In the second challenge we needed to develop a vision-based framework for learning the pick-and-place of regular and irregular-shaped objects in clutter. The third challenge revolved around developing a vision-based framework for learning the pick-and-place of regular and irregular-shaped objects in clutter with combination of prehensile and non-prehensile robotic manipulations. The major contributions arising from this work include:

Contribution I:

Developed and presented a comparative analysis of RL-based off-policy and on-policy temporal difference algorithms for industrial pick-and-place with non-visual sensing [16] [17].

This presented approach allows the agents to learn the pick-and-place task through reinforcement learning in such industrial environments where vision sensors are not viable due to certain factors. These factors can be shortage of installation space or high rate of dust or vibrations. In some scenarios, even wash-up from the water jets at the scene is one of the major reasons to avoid vision sensors. Most of the previously existing studies in this research area address the pick-and-place robotic manipulation through vision sensors only.

Therefore, this proposed algorithm addresses this research gap by enabling the agent to learn the pick-and-place task with the help of the ray-type proximity sensors instead of the vision sensors. We have trained and tested both off-policy (Q-learning) and on-policy (SARSA) temporal difference algorithms. The results show that Q-learning play better role in our proposed solution. Through this approach, the agent successfully learns to select the best suitable XYZ coordinates for the operation in accordance with the varying positions, orientations of objects on the conveyor belt and random conveyor belt speeds.

Contribution II:

Developed a DRL-based self-learning framework for industrial pick-and-place of regular and irregular shaped objects in clutter with the help of vision sensors [18].

This proposed framework enables the agent to learn and perform industrial pick-and-place of regular and irregular shaped objects in the clutter. There are various existing vision-based pick-and-place studies. But majority of them focus on first identifying the objects individually through different segmentation and singulation techniques resulting into additional overhead. Some other approaches require beforehand geometrical knowledge of the objects or domain specific knowledge. Our approach considers the whole workspace as one instead of dealing with objects individually using the pixelwise-parameterization technique and doesn't require any sort of beforehand domain specific knowledge or any geometrical data about the objects. We utilize the off-policy temporal difference Q-learning algorithm in our approach. Instead of dealing each object in the clutter individually, with the help of pixelwise-parameterization technique, success probabilities (Q-values) are generated for each and every pixel of the workspace. In the RL world, these success probabilities are known as expected future reward. This success probability of a pixel means the chances of success if the grasping action is carried out at that particular pixel location.

Contribution III:

Developed a DRL-based prehensile and non-prehensile robotic manipulation framework for industrial pick-and-place of regular and irregular objects in the clutter with the help of vision sensors.

This is an extended framework from our previous DRL-based pick-and-place approach. This extended approach enables the agent to learn and perform the industrial pick-and-place of regular and irregular shaped objects through sequence of prehensile and non-prehensile manipulations with the help of vision sensors. In existing literature, some approaches that tend to combine prehensile and non-prehensile motion can be witnessed. For instance, in such a way that prehensile motion (grasping) is performed during the non-prehensile motion or using non-prehensile motion (pushing) to move the object to certain locations for already known grasping policies. The common limitations of most of these existing approaches are their need of beforehand domain specific knowledge such as pretrained policies or handcrafted information and their requirement to perceive objects individually with the help of segmentation and singulation techniques. We address these both limitations with the help of pixelwise-parameterization technique as explained before.

Our approach addresses three actions grasping (prehensile), left-slide (non-prehensile) and right-slide (non-prehensile) by training three individual end-to-end memory-efficient variants of the DenseNet-121 and ResNet-101 for pixelwise function approximation together simultaneously. Any approach in the past using these networks without proper and comprehensive memory management unlike us, suffers from bottleneck situation at the GPU end as it experiences the quadratic feature growth with time. Our approach also enables the agents to learn bi-directional non-prehensile manipulation (left and right) as compared to the unidirectional and heuristic dependent previous approaches.

7.2 Future Work

Possible future extensions and improvements are as follows:

1. *Improving RL Algorithms*

This thesis utilized the deep Q-learning for the proposed approaches. In order to further improve the results other schemes/variants of Q-learning can also be trained and tested such as double Q-learning, dueling Q-learning and delayed Q-learning. The double Q-learning [128] can tackle the problem of overestimating the rewards by separating the action selection from action evaluation through modifying the Bellman equation. The results have shown successful overestimation reductions leading to comparatively better final policies at the end. The improvement in results has also been seen in the case of dueling Q-learning [129] where the Q-values are split into value and advantage functions and in the delayed Q-learning [130] where the estimates are delayed until there is a statistically significant sample of observations.

2. *Transition to Real-World*

Another future extension for the proposed approaches in this thesis can be the transition from the existing designed simulated environment to the real-life robotic arm in a physically designed industrial setup. The robotic operating system (ROS/ROS2) [131] can play a vital role in this transition.

3. *Other Considerations*

Some other improvements or modifications which can be considered involve extending the range of 3D designed objects in the experiments, real-life objects like cups, mugs, balls, etc., can be added to the list. In this thesis, sequential combination of robotic manipulation was achieved but it can be extended by adding more robotic manipulations such as stacking, rolling, etc.,

and attempting them in parallel fashion instead of sequential to unearth more efficient and expert schemes.

8 References

[1] S. Vaidya, P. Ambad, and S. Bhosle, “Industry 4.0 – A Glimpse,” *Procedia Manufacturing*, vol. 20, pp. 233–238, 2018, doi: https://doi.org/10.1016/j.promfg.2018.02.034.

[2] T. L.Friedman, “At Lunch, Donald Trump Gives Critics Hope,” *The New York Times*, Nov. 22, 2016. [Online]. Available: https://www.nytimes.com/2016/11/22/opinion/at-lunch-donald-trump-gives-critics-hope.html

[3] A. M. Optics Edmund, “Machine Vision Adapts to Harsh Environments.” https://www.photonics.com/Articles/Machine_Vision_Adapts_to_Harsh_Environments/a31905 (accessed Jun. 05, 2023).

[4] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: 10.1038/nature16961.

[5] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[6] OpenAI *et al.*, “Learning Dexterous In-Hand Manipulation,” *arXiv:1808.00177 [cs, stat]*, Jan. 2019, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1808.00177

[7] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, “Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost,” in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada: IEEE, May 2019, pp. 3651–3657. doi: 10.1109/ICRA.2019.8794102.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-End Training of Deep Visuomotor Policies,” *arXiv:1504.00702 [cs]*, Apr. 2016, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1504.00702

[9] I. Popov *et al.*, “Data-efficient Deep Reinforcement Learning for Dexterous Manipulation,” p. 12.

[10] M. Andrychowicz *et al.*, “Hindsight Experience Replay,” *arXiv:1707.01495 [cs]*, Feb. 2018, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1707.01495

[11] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine, “SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning,” *arXiv:1808.09105 [cs, stat]*, Jun. 2019, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1808.09105

[12] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Overcoming Exploration in Reinforcement Learning with Demonstrations,” *arXiv:1709.10089 [cs]*, Feb. 2018, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1709.10089

[13] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven Exploration by Self-supervised Prediction,” *arXiv:1705.05363 [cs, stat]*, May 2017, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1705.05363

[14] N. Savinov *et al.*, “Episodic Curiosity through Reachability,” *arXiv:1810.02274 [cs, stat]*, Aug. 2019, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1810.02274

[15] M. Riedmiller *et al.*, “Learning by Playing - Solving Sparse Reward Tasks from Scratch,” *arXiv:1802.10567 [cs, stat]*, Feb. 2018, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1802.10567

[16] M. B. Imtiaz, Y. Qiao, and B. Lee, “Implementing Robotic Pick and Place with Non-visual Sensing Using Reinforcement Learning,” in *2022 6th International Conference on Robotics, Control and Automation (ICRCA)*, Xiamen, China: IEEE, Feb. 2022, pp. 23–28. doi: 10.1109/ICRCA55033.2022.9828993.

[17] M. Imtiaz, Y. Qiao, and B. Lee, “Comparison of Two Reinforcement Learning Algorithms for Robotic Pick and Place with Non-Visual Sensing,” *International Journal of Mechanical Engineering and Robotics Research*, pp. 526–535, Jan. 2021, doi: 10.18178/ijmerr.10.10.526-535.

[18] M. B. Imtiaz, Y. Qiao, and B. Lee, “Prehensile Robotic pick-and-place in clutter with Deep Reinforcement Learning,” in *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, Prague, Czech Republic: IEEE, Jul. 2022, pp. 1–6. doi: 10.1109/ICECET55527.2022.9873426.

[19] M. B. Imtiaz, Y. Qiao, and B. Lee, “Prehensile and Non-Prehensile Robotic Pick-and-Place of Objects in Clutter Using Deep Reinforcement Learning,” *Sensors*, vol. 23, no. 3, Art. no. 3, Jan. 2023, doi: 10.3390/s23031513.

[20] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” p. 352.

[21] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992, doi: 10.1007/BF00992698.
[22] G. A. Rummery and M. Niranjan, “ON-LINE Q-LEARNING USING CONNECTIONIST SYSTEMS,” p. 22.
[23] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” *CoRR*, vol. abs/1312.5602, 2013, [Online]. Available: http://arxiv.org/abs/1312.5602
[24] G. Brockman *et al.*, “OpenAI Gym.” arXiv, 2016. doi: 10.48550/ARXIV.1606.01540.
[25] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, “Evolution Strategies as a Scalable Alternative to Reinforcement Learning.” arXiv, 2017. doi: 10.48550/ARXIV.1703.03864.
[26] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, “Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation.” arXiv, 2017. doi: 10.48550/ARXIV.1708.05144.
[27] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, “A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4111–4117. doi: 10.1109/IROS.2013.6696945.
[28] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033. doi: 10.1109/IROS.2012.6386109.
[29] N. Koenig and A. Howard, “Design and use paradigms for Gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004, pp. 2149–2154 vol.3. doi: 10.1109/IROS.2004.1389727.
[30] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, “Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator,” in *Simulation, Modeling, and Programming for Autonomous Robots*, N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, and O. von Stryk, Eds., in Lecture Notes in Computer Science, vol. 6472. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 51–62. doi: 10.1007/978-3-642-17319-6_8.
[31] E. Coumans and Y. Bai, *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. 2016. [Online]. Available: http://pybullet.org
[32] “RaiSim v1.1.5 — raisim 1.1.5 documentation.” https://raisim.com/ (accessed Sep. 28, 2022).
[33] A. Sahbani, S. El-Khoury, and P. Bidaud, “An overview of 3D object grasp synthesis algorithms,” *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012, doi: https://doi.org/10.1016/j.robot.2011.07.016.
[34] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-Driven Grasp Synthesis—A Survey,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014, doi: 10.1109/TRO.2013.2289018.
[35] J. Mahler *et al.*, “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics,” *ArXiv*, vol. abs/1703.09312, 2017.
[36] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, “Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE Press, 2018, pp. 1–8. doi: 10.1109/ICRA.2018.8460887.
[37] D. Morrison, J. Leitner, and P. Corke, “Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach,” in *Robotics: Science and Systems XIV*, Robotics: Science and Systems Foundation, Jun. 2018. doi: 10.15607/RSS.2018.XIV.021.
[38] J. Redmon and A. Angelova, “Real-Time Grasp Detection Using Convolutional Neural Networks.” 2015.
[39] S. Kumra and C. Kanan, “Robotic grasp detection using deep convolutional neural networks,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 769–776. doi: 10.1109/IROS.2017.8202237.
[40] S. James *et al.*, “Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks.” arXiv, 2018. doi: 10.48550/ARXIV.1812.07252.
[41] B. Siciliano and O. Khatib, Eds., *Springer handbook of robotics*. Berlin: Springer, 2008.
[42] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, “Grasp Pose Detection in Point Clouds,” *The International Journal of Robotics Research*, vol. 36, no. 13–14, pp. 1455–1473, 2017, doi: 10.1177/0278364917735594.

[43] F. Spenrath and A. Pott, “Gripping Point Determination for Bin Picking Using Heuristic Search,” *Procedia CIRP*, vol. 62, pp. 606–611, 2017, doi: https://doi.org/10.1016/j.procir.2016.06.015.
[44] T. Hodan, J. Matas, and S. Obdrzálek, “On Evaluation of 6D Object Pose Estimation,” in *ECCV Workshops*, 2016.
[45] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley, “Symmetry Aware Evaluation of 3D Object Detection and Pose Estimation in Scenes of Many Parts in Bulk,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 2209–2218. doi: 10.1109/ICCVW.2017.258.
[46] S. Hinterstoisser *et al.*, “Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes,” in *Computer Vision – ACCV 2012*, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562.
[47] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley, “Defining the Pose of Any 3D Rigid Object and an Associated Distance,” *International Journal of Computer Vision*, vol. 126, no. 6, pp. 571–596, Nov. 2017, doi: 10.1007/s11263-017-1052-4.
[48] M. El-Shamouty, K. Kleeberger, A. Lämmle, and M. F. Huber, “Simulation-driven machine learning for robotics and automation,” *tm - Technisches Messen*, vol. 86, pp. 673–684, 2019.
[49] M. Palzkill, “Heuristisches Suchverfahren zur Objektlageerkennung aus Punktewolken für industrielle Zuführsysteme,” p. 164.
[50] T. Ledermann, *Partikel-Schwarm-Optimierung zur Objektlageerkennung in Tiefendaten*. in ISW/IPA Forschung und Praxis, no. 191. Heimsheim: Jost-Jetter, 2012.
[51] K. Kleeberger, C. Landgraf, and M. F. Huber, “Large-scale 6D Object Pose Estimation Dataset for Industrial Bin-Picking,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2573–2578. doi: 10.1109/IROS40897.2019.8967594.
[52] K. Kleeberger and M. F. Huber, “Single Shot 6D Object Pose Estimation.” arXiv, 2020. doi: 10.48550/ARXIV.2004.12729.
[53] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, “Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects.” arXiv, 2018. doi: 10.48550/ARXIV.1809.10790.
[54] Z. Dong *et al.*, “PPR-Net:Point-wise Pose Regression Network for Instance Segmentation and 6D Pose Estimation in Bin-picking Scenarios,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1773–1780. doi: 10.1109/IROS40897.2019.8967895.
[55] B. Tekin, S. N. Sinha, and P. Fua, “Real-Time Seamless Single Shot 6D Object Pose Prediction.” arXiv, 2017. doi: 10.48550/ARXIV.1711.08848.
[56] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, “Implicit 3D Orientation Learning for 6D Object Detection from RGB Images.” arXiv, 2019. doi: 10.48550/ARXIV.1902.01275.
[57] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, “SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again.” arXiv, 2017. doi: 10.48550/ARXIV.1711.10006.
[58] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An Accurate O(n) Solution to the PnP Problem,” *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, Feb. 2009, doi: 10.1007/s11263-008-0152-6.
[59] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30. doi: 10.1109/IROS.2017.8202133.
[60] T. Hodan *et al.*, “BOP: Benchmark for 6D Object Pose Estimation.” arXiv, 2018. doi: 10.48550/ARXIV.1808.08319.
[61] “SIXD Challenge 2017.” http://cmp.felk.cvut.cz/sixd/challenge_2017/ (accessed Oct. 01, 2022).
[62] “Competition at IROS 2019.” https://www.bin-picking.ai/en/competition.html (accessed Oct. 01, 2022).
[63] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.” arXiv, 2017. doi: 10.48550/ARXIV.1706.02413.
[64] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method*. in Information Science and Statistics. New York, NY: Springer, 2004. doi: 10.1007/978-1-4757-4321-0.
[65] A. Mousavian, C. Eppner, and D. Fox, “6-DOF GraspNet: Variational Grasp Generation for Object Manipulation.” arXiv, 2019. doi: 10.48550/ARXIV.1905.10520.

[66] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, *Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection*. 2016.
[67] J. Mahler *et al.*, "Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1957–1964. doi: 10.1109/ICRA.2016.7487342.
[68] J. Mahler *et al.*, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019, doi: 10.1126/scirobotics.aau4984.
[69] V. Satish, J. Mahler, and K. Goldberg, "On-Policy Dataset Synthesis for Learning Robot Grasping Policies Using Fully Convolutional Deep Networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1357–1364, 2019, doi: 10.1109/LRA.2019.2895878.
[70] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3304–3311. doi: 10.1109/ICRA.2011.5980145.
[71] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger." arXiv, 2016. doi: 10.48550/ARXIV.1612.08242.
[72] C. Szegedy, A. Toshev, and D. Erhan, "Deep Neural Networks for Object Detection," in *Advances in Neural Information Processing Systems*, C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/f7cade80b7cc92b991cf4d2806d6bd78-Paper.pdf
[73] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.
[74] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4–5, pp. 705–724, 2015, doi: 10.1177/0278364914549607.
[75] "cornell_grasp." https://www.kaggle.com/datasets/oneoneliu/cornell-grasp (accessed Oct. 02, 2022).
[76] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A Large Scale Dataset for Robotic Grasp Detection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3511–3516. doi: 10.1109/IROS.2018.8593950.
[77] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, vol. 39, no. 2–3, pp. 183–201, 2020, doi: 10.1177/0278364919859066.
[78] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics." arXiv, 2019. doi: 10.48550/ARXIV.1903.11239.
[79] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. A. Funkhouser, "Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning," *CoRR*, vol. abs/1803.09956, 2018, [Online]. Available: http://arxiv.org/abs/1803.09956
[80] L. Berscheid, P. Meißner, and T. Kröger, "Robot Learning of Shifting Objects for Grasping in Cluttered Environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 612–618. doi: 10.1109/IROS40897.2019.8968042.
[81] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep Reinforcement Learning for Vision-Based Robotic Grasping: A Simulated Comparative Evaluation of Off-Policy Methods." arXiv, 2018. doi: 10.48550/ARXIV.1802.10264.
[82] D. Kalashnikov *et al.*, "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation." arXiv, 2018. doi: 10.48550/ARXIV.1806.10293.
[83] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4–5, pp. 421–436, 2018, doi: 10.1177/0278364917710318.
[84] K. Bousmalis *et al.*, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE Press, 2018, pp. 4243–4250. doi: 10.1109/ICRA.2018.8460875.
[85] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the Wild:Learning 6DoF Closed-Loop Grasping from Low-Cost Demonstrations." arXiv, 2019. doi: 10.48550/ARXIV.1912.04344.
[86] M. Gualtieri, A. ten Pas, and R. W. Platt, "Category Level Pick and Place Using Deep Reinforcement Learning," *ArXiv*, vol. abs/1707.05615, 2017.
[87] M. Gualtieri, A. ten Pas, and R. Platt, *Pick and Place Without Geometric Object Models*. 2018.

[88] D. Prattichizzo and J. C. Trinkle, “Grasping,” in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 671–700. doi: 10.1007/978-3-540-30301-5_29.
[89] J. Weisz and P. K. Allen, “Pose error robust grasping from contact wrench space metrics,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 557–562. doi: 10.1109/ICRA.2012.6224697.
[90] A. Rodriguez, M. T. Mason, and S. Ferry, “From caging to grasping,” *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, Jun. 2012, doi: 10.1177/0278364912442972.
[91] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, “The Columbia grasp database,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1710–1716. doi: 10.1109/ROBOT.2009.5152709.
[92] A. Zeng *et al.*, “Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore: IEEE, May 2017, pp. 1386–1383. doi: 10.1109/ICRA.2017.7989165.
[93] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, “3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions,” in *CVPR*, 2017.
[94] A. Zeng *et al.*, “Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching.” arXiv, 2017. doi: 10.48550/ARXIV.1710.01330.
[95] L. Pinto and A. Gupta, “Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours,” *CoRR*, vol. abs/1509.06825, 2015, [Online]. Available: http://arxiv.org/abs/1509.06825
[96] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, “High precision grasp pose detection in dense clutter,” *arXiv:1603.01564 [cs]*, Jun. 2017, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1603.01564
[97] L. Pinto and A. Gupta, “Learning to Push by Grasping: Using multiple tasks for effective learning,” *CoRR*, vol. abs/1609.09025, 2016, [Online]. Available: http://arxiv.org/abs/1609.09025
[98] S. Goyal, A. Ruina, and J. Papadopoulos, “Planar sliding with dry friction Part 1. Limit surface and moment function,” *Wear*, vol. 143, no. 2, pp. 307–330, 1991, doi: https://doi.org/10.1016/0043-1648(91)90104-3.
[99] M. T. Mason, “Mechanics and Planning of Manipulator Pushing Operations,” *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986, doi: 10.1177/027836498600500303.
[100] M. Bauzá and A. Rodriguez, “A probabilistic data-driven model for planar pushing,” *CoRR*, vol. abs/1704.03033, 2017, [Online]. Available: http://arxiv.org/abs/1704.03033
[101] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, “More than a million ways to be pushed. A high-fidelity experimental dataset of planar pushing,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 30–37. doi: 10.1109/IROS.2016.7758091.
[102] T. Mericli, M. M. Veloso, and H. L. Akin, “Push-Manipulation of Complex Passive Mobile Objects using Experimentally Acquired Motion Models.” Carnegie Mellon University, Mar. 2015. doi: 10.1184/R1/6608750.v1.
[103] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini, “A Vision-Based Learning Method for Pushing Manipulation,” p. 8.
[104] J. Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason, “A convex polynomial force-motion model for planar sliding: Identification and application,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 372–377. doi: 10.1109/ICRA.2016.7487155.
[105] M. R. Dogar and S. S. Srinivasa, “A Planning Framework for Non-Prehensile Manipulation under Clutter and Uncertainty,” *Auton Robot*, vol. 33, no. 3, pp. 217–236, Oct. 2012, doi: 10.1007/s10514-012-9306-z.
[106] I. Clavera, D. Held, and P. Abbeel, “Policy Transfer via Modularity,” p. 9.
[107] D. Omrcen, C. Boge, T. Asfour, A. Ude, and R. Dillmann, “Autonomous acquisition of pushing actions to support object grasping with a humanoid robot,” in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, Paris: IEEE, Dec. 2009, pp. 277–283. doi: 10.1109/ICHR.2009.5379566.
[108] A. Boularias, J. A. Bagnell, and A. Stentz, “Learning to Manipulate Unknown Objects in Clutter by Reinforcement,” in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, in AAAI’15. AAAI Press, 2015, pp. 1336–1342.

[109] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, “Automatic grasp planning using shape primitives,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, Taipei, Taiwan: IEEE, 2003, pp. 1824–1829. doi: 10.1109/ROBOT.2003.1241860.
[110] M. Tokic and G. Palm, “Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax,” in *KI 2011: Advances in Artificial Intelligence*, J. Bach and S. Edelkamp, Eds., in Lecture Notes in Computer Science, vol. 7006. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 335–346. doi: 10.1007/978-3-642-24455-1_33.
[111] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, “Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization,” in *Robotics: Science and Systems IX*, Robotics: Science and Systems Foundation, Jun. 2013. doi: 10.15607/RSS.2013.IX.031.
[112] R. Diankov and J. J. Kuffner, “OpenRAVE: A Planning Architecture for Autonomous Robotics,” 2008.
[113] I. A. Sucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012, doi: 10.1109/MRA.2012.2205651.
[114] J. J. Kuffner and S. M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, San Francisco, CA, USA: IEEE, 2000, pp. 995–1001. doi: 10.1109/ROBOT.2000.844730.
[115] Z. Zou, J. Han, and M. Zhou, “Research on the inverse kinematics solution of robot arm for watermelon picking,” in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu: IEEE, Dec. 2017, pp. 1399–1402. doi: 10.1109/ITNEC.2017.8285026.
[116] L. Cruz, D. Lucio, and L. Velho, “Kinect and RGBD Images: Challenges and Applications,” in *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials*, 2012, pp. 36–49. doi: 10.1109/SIBGRAPI-T.2012.13.
[117] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *arXiv:1608.06993 [cs]*, Jan. 2018, Accessed: Apr. 29, 2021. [Online]. Available: http://arxiv.org/abs/1608.06993
[118] P. J. Huber, “Robust Estimation of a Location Parameter,” *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.
[119] S. Ruder, “An overview of gradient descent optimization algorithms.” arXiv, Jun. 15, 2017. Accessed: Oct. 21, 2022. [Online]. Available: http://arxiv.org/abs/1609.04747
[120] F. Lamiraux and J. Mirabel, “Prehensile Manipulation Planning: Modeling, Algorithms and Implementation,” *IEEE Trans. Robot.*, pp. 1–19, 2021, doi: 10.1109/TRO.2021.3130433.
[121] D. Serra, “Robot Control for Nonprehensile Dynamic Manipulation Tasks,” p. 10.
[122] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.
[123] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.
[124] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, “Memory-Efficient Implementation of DenseNets,” *CoRR*, vol. abs/1707.06990, 2017, [Online]. Available: http://arxiv.org/abs/1707.06990
[125] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL: IEEE, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.
[126] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385 [cs]*, Dec. 2015, Accessed: Apr. 29, 2021. [Online]. Available: http://arxiv.org/abs/1512.03385
[127] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.
[128] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, in AAAI’16. AAAI Press, 2016, pp. 2094–2100.
[129] Z. Wang, N. de Freitas, and M. Lanctot, “Dueling Network Architectures for Deep Reinforcement Learning,” *CoRR*, vol. abs/1511.06581, 2015, [Online]. Available: http://arxiv.org/abs/1511.06581

[130] B. Han, Z. Ren, Z. Wu, Y. Zhou, and J. Peng, “Off-Policy Reinforcement Learning with Delayed Rewards.” arXiv, 2021. doi: 10.48550/ARXIV.2106.11854.
[131] Stanford Artificial Intelligence Laboratory et al., “Robotic Operating System.” May 23, 2018. [Online]. Available: https://www.ros.org