JCSDA, Vol. 02, No. 01, 29–62
DOI: 10.69660/jcsda.02012503
ISSN 2959-6912

Deep Reinforcement Learning Based Synergies Pushing and Grasping Policies in Cluttered Scene Using UR5 Robot

Birhanemeskel A. Shiferaw*, Ramasamy Srinivasan and Tayachew Fikire

Department of Electromechanical Engineering,
Artificial Intelligence and Robotics Center of Excellence,
Addis Ababa Science and Technology University, Addis Ababa, Ethiopia
**corresponding author: birmeskelal@gmail.com*

This paper presents deep reinforcement learning-based synergy between pushing and grasping systems to improve the grasping performance of the UR5 robot in a cluttered scene. In robotic manipulations, grasping an object in a clutter is fundamental yet a challenging activity for industrial applications. This is because numerous studies focused on improving grasping performance in cluttered environments using either a grasping-only policy or pushing and grasping without incorporating a pushing reward. Additionally, some research has been limited to using similar objects, such as cubes. This paper formulated the mathematical modeling of the universal robot manipulator. The proposed model involves training two fully-connected convolutional neural networks that transfer visual observations of the scene to a dense pixel-wise sample of end-effector orientations and positions for each pushing and grasping action. A fixed RGB-D camera is used to take the raw images of the scene and generate a heightmap image. Before feeding the heightmap image to the fully convolutional network, it is rotated by 36 different angles to generate 36 pixel-wise Q-value predictions. Both pushing and grasping networks are self-supervised by trial and error from experience and are trained together in a deep Q-learning algorithm. Successful grasps have a reward of 1, while successful pushes have a 0.5 reward value. But unsuccessful actions have a reward of 0 value. The proposed policy learns pushing motions to improve future grasping in a cluttered scene. The experiment demonstrates that the proposed model can successfully grasp objects with an 87% grasp success rate while grasping only policy, no-reward for pushing policy, and stochastic gradient without momentum is 60%, 71%, and 79% respectively. Further, it has been demonstrated that the proposed model is capable of generalizing to randomly arranged cluttered objects, challenging arrangements, and novel objects.

Keywords: Deep reinforcement learning, synergy, robotic grasping, cluttered scene, Q-value, UR5.

1. Introduction

Robots were created to aid or replace humans by performing tedious and risky jobs that humans either do not want to undertake or are unable to execute due to physical limits imposed by harsh conditions [1]. Robotic arms are automated electromechanical devices designed to carry out specific tasks [2]. Applications of robotics are widely available in our daily life ranging from small household robots to large manufacturing industries.

The Universal Robots company product, UR5 a six degree of freedom is widely used in most research areas due to their lightweight, speed, easy to program, flexibility, and safety. All of its six revolute joints contribute to the transformational and rotational movements of its end effector [3].

Artificial intelligence (AI) has become a critical component in the subject of robotics. The definition of AI varies depending on its application and field, but [4] defines it as the creation of an intelligent agent that can interact with the environment and take actions to maximize its success, in which the agent acts intelligently to reach the optimal result.

A biologically inspired engineering model, artificial neural network (ANN) consists of numerous single units called artificial neurons that are coupled with coefficients (weights) to form the neural structure [5]. It contains devices with many inputs and one or more outputs. An artificial neural network is consisting of a large number of fundamental processing pieces that are linked and layered together.

Deep neural network learning is a well-known field in the family of machine learning, with its excellent achievement in a variety of domains ranging from classical computer vision tasks to many other practical applications. Deep-learning based methods have achieved comparable to, and in some cases outperforming human expert performance. Deep learning has enhanced data processing, making decisions, data analyzing and manipulation [1].

A type of machine learning in which the UR5 learns optimal behavior through experience by trial-and-error interactions with the environment in order to maximize its performance is called reinforcement learning. A reinforcement learning UR5 is not informed which actions to take, instead, it must try them all to see which ones offer most reward. It chooses behaviors based on previous experiences as well as tries now options with the goal of maximizing the total reward [6].

The hybrid between deep learning and reinforcement learning is deep reinforcement learning. The robot learns from its actions similar to the way humans can learn from experience. The robot learns complex features by trial-and-error interactions with the environment so as to optimize its performance over time.

A robotics problem is formalized by defining a state and action space, and the dynamics which describe how actions influence the state of the system. The state space includes internal states of the robot as well as the state of the world that is intended to be controlled. Quite often, the state is not directly observable–instead, the robot is equipped with sensors, which provide observations that can be used to infer the state. The goal may be defined either as a target state to be achieved, or as a reward function to be maximized. We want to find a controller, (known as a policy in reinforcement learning), that maps states to actions in a way that maximizes the reward when executed [7].

2. Related works

For effective and efficient object manipulation, an industrial robot manipulator must be able to sense and interact with its surroundings. Despite extensive researches, robotic grasping in a cluttered environment remains a difficult challenge, as do many other robotic manipulations. Using deep reinforcement learning, several distinct approaches to grasping in a cluttered environment have been developed in

recent years.

A scalable visual grasping system [8] proposes a robotic grasping strategy based on the model-free deep reinforcement learning, named Deep Reinforcement Grasp Policy (DRGP). The proposed system requires little training time and limited simple objects in simulation but generalizes well to novel objects in a real-world environment. A perception network employs a convolutional neural network to translate visual data to grasp action in which dense pixel-wise Q-values indicate the position and orientation of the robot's primitive action. Grasp success rate reaches 93% on a single arrangement unknown object, 70% on six unknown objects in a crowded situation, and 62% on seven objects in crowded situations. The methodology is based on only grasping policy as a result, it fails to perform well in a cluttered environment. With out a non-prehensile action (pushing), it is difficult to grasp an object in a dense scene.

In [9], a robotic grasp-to-place method that can grasp objects in sparse and cluttered scene was presented. The main achievement of this study is that it can handle both picking and placing using raw RGB-D images and an explicit framework. The RGB-D camera was utilized to produce heightmaps at the robot grasp-workspace by capturing RGB-D images of the scene and 3D point cloud information. The heightmap is rotated by 36 different angles before being fed into a dense connected convolutional network (DenseNet121) to produce 36 pixel-wise Q-value maps predictions. The suggested model has 77.4% grasp efficiency and 74.3% grasp success rate, respectively and 98.42% place success rate. The paper proposes a pick and place robot in a cluttered environment, but it failed to provide a solution for better grasping an object in the cluttered environment. The grasping process is not enhanced by the non-prehensile (push) action which resulted in low grasping efficiency and grasp success rate of an object. The model also failed to provide tests for novel objects.

In [10], a deep reinforcement learning-based technique for tackling the robotic grasping issue using Visio-motor feedback was proposed. The suggested system is a visual servoing method that perceives the environment including the objects of interest using a multi-view camera setup. The model is made up of observed data from the overhead depth camera and wrist RGB camera, as well as current motor position, all of which are sent into Grasp-Q-Network, which yields grasp success probability. The model tested in both a Baxter Gazebo simulated environment and on a real robot. Even though adopting a multi-view model increases grasping accuracy in comparison to a single-view model, the training experiment was performed only with three regular shaped colored objects; sphere, cylinder, and cube. This study highlights the efficacy of reinforcement learning in sparse settings, though it lacks detailed exploration in highly cluttered scenarios.

The paper presented in [11] used both Schedule for Positive Task (SPOT) reward and the schedule for Positive Task (SPOT)-Q reinforcement learning algorithms, which resulted in efficient learning multi-step block manipulation tasks in both simulation and real-world environments. The Schedule for Positive Task (SPOT) incen-

tive system works on the principle that actions that advance overall task progress rewarded proportionally to the amount of progress made while actions that reverse progress are not rewarded. By recording stack height or row length, SPOT can calculate task progress (TP). The heightmap is rotated by 16 distinct angles which is then fed into a Dense Network (DenseNet-121) to yield predictions of 16 pixel-wise Q-value maps. The paper achieved a grasp success rate of 76% in simulation and 69% in real training on stack of 4 cubes. The proposed system failed to give a reward for pushing action that leads to lower grasp success. As successful pushing actions are not rewarded, the contribution of pushing for improving grasping is minimum. The proposed system failed to give a reward for successful pushing action that leads to a lower grasp success. Without a push reward, the robot will not learn push policies.

To grasp unseen target object in a clutter trained by self-supervision in simulation was presented in [12]. The method employed three fully connected convolutional neural networks, DenseNet-121, for feature extraction of color, depth and the target object, which demands high computational cost. It has low grasping success rate and performs more motion. This is because it seeks to grasp the object directly without separating the objects. It tends to push the cluttered environment after multiple continuous grasp failures. Grasping is also limited to a target, goal oriented single object.

The paper presented in [13] employs deep Q-learning for gripping policies and pushing for applications involving crowded. Through model-free deep reinforcement learning, this research proved that it is possible to uncover and learn synergies between push and pick in a clutter. Before feeding the heightmap image into the DenseNet-121 network to generate 16 Q-value predictions, it is rotated in 16 different angles. The model grasping performance for the developed system achieved 83% over 2500 episodes. The paper's key weakness is that it repeatedly pushes an object out of the workspace because pixel-wise depth heightmap image differencing (PDD) approach is not employed to assure success push. A second limitation is that the system has trained and tested with only blocks.

3. Material and methods

3.1. *Experimental setup*

The experiments were conducted using a UR5 robotic arm from Universal Robots, equipped with a two-finger parallel gripper. An RGB-D camera was fixed in the environment to capture the scene and generate height map images. These images were used as inputs for the neural networks.

The experimental setup involves using a UR5 robotic arm to test and validate the proposed robotic manipulation model. The UR5 robot is a versatile and widely used industrial robot known for its precision and flexibility, making it ideal for tasks involving grasping and manipulation in cluttered environments. The setup is designed to evaluate the model's performance in various scenarios, including

randomly arranged clutter, well-ordered configurations, and interactions with novel objects.

Hardware components

(1) UR5 Robotic Arm:
Model: Universal Robots UR5.
Degrees of Freedom: 6.
Payload: 5 kg.
Reach: 850 mm.
Repeatability: ±0.1 mm.
(2) End Effector:
Type: RG2 parallel jaw gripper.
Grip Force: 20-235 N.
Grip Stroke: 0-85 mm
(3) Vision System:
Camera: Intel RealSense Depth Camera D435.
Resolution: 640 x 480 at 30 fps.
Depth Range: 0.2-10 m.
(4) Computing System:
Processor: Intel Core i7.
GPU: NVIDIA Quadro P2000 Intel(R) Xeon(R)
RAM: 32 GB.

Software Components

(1) Operating System: Ubuntu 18.04 LTS.
(2) Robotics Middleware: Virtual Robot Experimentation Platform (V-REP).
(3) Control Interface: CoppeliaSim VR interface with the UR5 robot.
(4) Simulation Environment: CoppeliaSim simulation.
(5) Machine Learning Framework: PyTorch for model training and deployment.

Experimental Procedure

(1) Environment Setup:

- The robot is placed in a controlled test area with a flat, non-reflective surface to minimize visual noise.
- Objects for manipulation are randomly placed within the robot's workspace to simulate cluttered environments.
- A mix of known and novel objects is used to evaluate the model's adaptability.

(2) Calibration:

- The vision system is calibrated using standard calibration techniques to ensure accurate depth perception.

Fig. 1. UR5 robotic arm [14]

- The end effector is calibrated to ensure precise gripping force and stroke control.

(3) Task Execution:

- The robot is programmed to perform a series of grasping tasks, starting with randomly arranged clutter.
- The tasks are repeated with well-ordered object configurations and novel objects to test the model's adaptability and performance consistency.
- Each task involves identifying the object, planning a grasp, and executing the grasp.

3.2. *UR5 mathematical modeling and deep reinforcement algorithm*

3.2.1. *UR5 mathematical modeling*

Six revolute joints make up the UR5 robotic arm. Base, Shoulder, Elbow, Wrist1, Wrist2, and Wrist3 are the names of these joints. The shoulder and elbow joints rotate perpendicular to the base joint. Long linkages connect the shoulders, elbows, and base joints [3].

3.2.2. *D-H Representation of Forward Kinematic Equations of UR5*

The UR5 robot's kinematics can be studied using the D-H representation. To efficiently control the end-effector with regard to the base, the relationship between the

coordinate frames attached to the end-effector and the base of the UR5 robot must be discovered. This is accomplished by recursively describing transformations of the coordinates between the coordinate frames connected to each link and yielding an overall description.

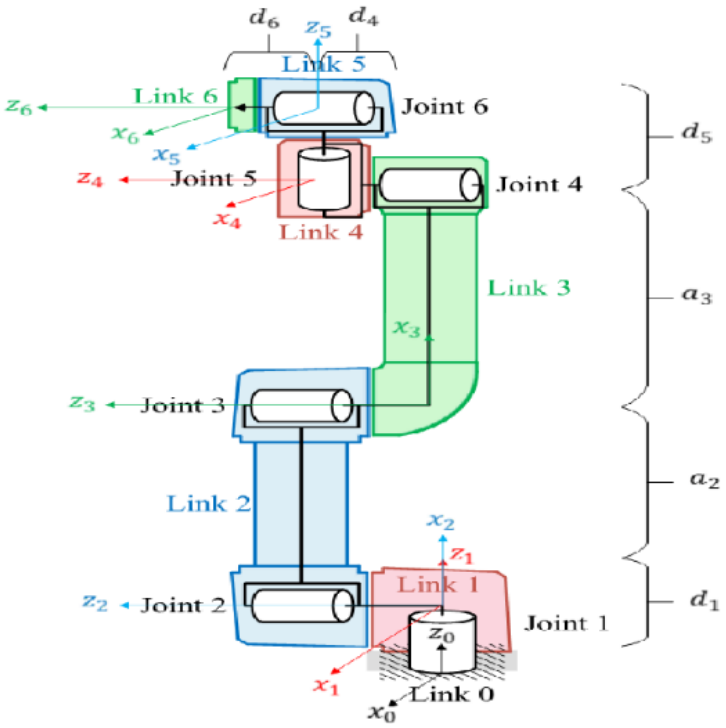


Fig. 2. Schematic and frames assignment of UR5

Table 1. D-H parameters of the UR5

Link, i	a_i	α_i	d_i	θ_i
1	*0*	$\pi/2$	*0.0891*	θ_1
2	*0.425*	*0*	*0*	θ_2
3	*0.39225*	*0*	*0*	θ_3
4	*0*	$\pi/2$	*0.10915*	θ_4
5	*0*	$-\pi/2$	*0.09456*	θ_5
6	-	-	*0.0823*	θ_6

Using the definition of the transformation matrix of a robotic arm, the transformation matrix from the base to the end effector of the UR5 robot is in the form

of:

$$\begin{bmatrix} {}^0_1T(\theta_1) & {}^1_2T(\theta_2) & {}^2_3T(\theta_3) \\ {}^3_4T(\theta_4) & {}^4_5T(\theta_5) & {}^5_6T(\theta_6) \end{bmatrix} = \left[\begin{bmatrix} c1 & 0 & s1 & 0 \\ s1 & 0 & -c1 & 0 \\ 0 & 1 & 0 & 0.089 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c2 & -s2 & 0 & -0.425c2 \\ s2 & c2 & 0 & -0.425s2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c3 & -s3 & 0 & -0.39c3 \\ s2 & c2 & 0 & -0.39s3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right. \\ \left. \begin{bmatrix} c4 & 0 & s4 & 0 \\ s4 & 0 & -c4 & 0 \\ 0 & 1 & 0 & 0.109 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c5 & 0 & -s5 & 0 \\ s5 & 0 & c5 & 0 \\ 0 & -1 & 0 & 0.09465 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c6 & -s6 & 0 & 0 \\ s6 & c6 & 0 & 0 \\ 0 & 0 & 1 & 0.0823 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right] \quad (1)$$

So, the transformation of the UR5 robot from the base to the end-effector is the post-product of all the six transformation matrices.

$${}^0_6T(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = {}^0_1T(\theta_1) * {}^1_2T(\theta_2) * {}^2_3T(\theta_3) * {}^3_4T(\theta_4) * {}^4_5T(\theta_5) * {}^5_6T(\theta_6) = \begin{bmatrix} \mathrm{n_x} & \mathrm{o_x} & \mathrm{a_x} & \mathrm{p_x} \\ \mathrm{n_y} & \mathrm{o_y} & \mathrm{a_y} & \mathrm{p_y} \\ \mathrm{n_z} & \mathrm{o_z} & \mathrm{a_z} & \mathrm{p_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The forward kinematics of the UR5 robot orientation would be obtained from the first three columns of the ${}^0_6T$ in Eq. (2) as:

$$\begin{bmatrix} \mathrm{n_x} & \mathrm{o_x} & \mathrm{a_x} \\ \mathrm{n_y} & \mathrm{o_y} & \mathrm{a_y} \\ \mathrm{n_z} & \mathrm{o_z} & \mathrm{a_z} \end{bmatrix} = \begin{bmatrix} \mathrm{s_1}s_5\mathrm{c_6} + c_1\mathrm{c_{234}}c_5\mathrm{c_6} + c_1\mathrm{s_{234}}s_6 & -c_1\mathrm{s_{234}}c_6 - \mathrm{s_1}s_5s_6 - c_1\mathrm{c_{234}}\mathrm{c_5}s_6 & \mathrm{s_1c_5} - c_1\mathrm{c_{234}} \\ s_1\mathrm{c_{234}}\mathrm{c_5}\mathrm{c_6} - \mathrm{c_1}s_5\mathrm{c_6} - s_1\mathrm{s_{234}}s_6 & -s_1\mathrm{s_{234}}c_6 - s_1\mathrm{c_{234}}\mathrm{c_5}s_6 + \mathrm{c_1}s_5s_6 & -\mathrm{c_1c_5} - s_1\mathrm{c_{234}} \\ s_{234}\mathrm{c_5}\mathrm{c_6} + c_{234}s_6 & \mathrm{c_{234}}c_6 - s_{234}\mathrm{c_5}\mathrm{s_6} & -s_{234}\mathrm{s_5} \end{bmatrix} \quad (3)$$

And the forward kinematics of the robot position would easily be obtained from the fourth column of the ${}^0_6T$ in Eq. (2) as:

$$\begin{cases} \mathrm{p_x}=\mathrm{d_5c_1s_{234}}+\mathrm{d_4s_1}-\mathrm{d_6c_1s_{234}}+\mathrm{a_2c_1c_2}+\mathrm{d_6c_5s_1}+\mathrm{a_3c_1c_2c_3}-\mathrm{a_3c_1s_2s_3} \\ \mathrm{p_y}=\mathrm{d_5s_1s_{234}}-\mathrm{d_4c_1}-\mathrm{d_6s_1c_{234}}+\mathrm{a_2c_2c_1}-\mathrm{d_6c_1s_5}+\mathrm{a_3c_2c_3s_1}-\mathrm{a_3s_1s_2s_3} \\ \mathrm{p_z}=\mathrm{d_1}-\mathrm{d_6s_5s_{234}}+\mathrm{a_3s_{23}}+\mathrm{a_2s_2}-\mathrm{d_5c_{234}} \end{cases} \quad (4)$$

Where, $\mathrm{s_{234}}$ stands for the $\sin(\theta_2+\theta_3+\theta_4)$ and $\mathrm{c_{234}}$ stands for the $\cos(\theta_2+\theta_3+\theta_4)$.

3.2.3. *Inverse Kinematics of the UR5*

The goal of inverse kinematics is to discover the joint angles vectors that will cause the end effector to achieve a particular goal state. The inverse kinematic equation calculates the joint angles $\theta_1 - \theta_6$ based on a desired position and orientation of the end-effector, specified as the transformation ${}^0_6T$. To describes position and orientation of the UR5 robot manipulator in the inverse kinematics solution, all angles are restricted as $\left[\theta_1^i, \ldots, \theta_6^i\right]^T \in [0;\ 2\pi)$, such that satisfies Eq (2). In this work, geometric method was used to solve the inverse kinematics of the UR5 robot.

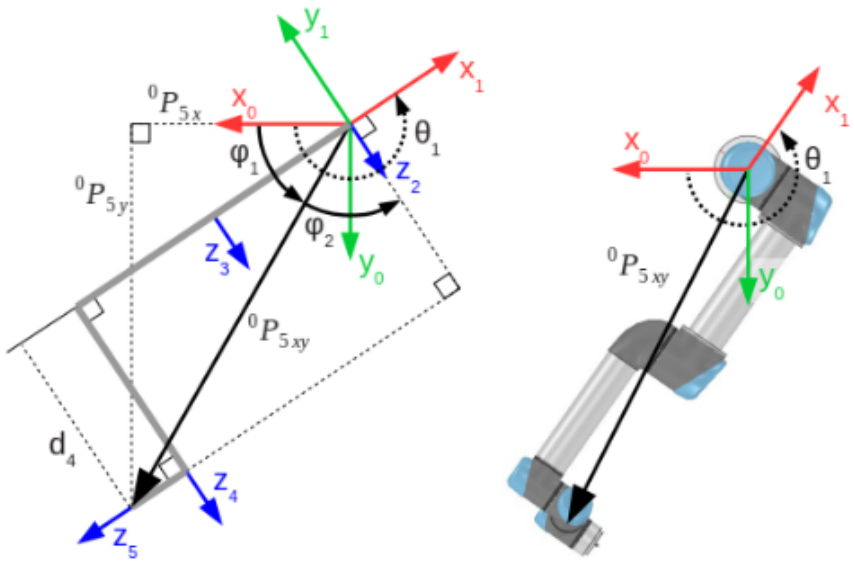


Fig. 3. Geometry of the UR5 for finding θ_1

The wrist, $P5$, is considered from frame 0 and frame 1, respectively, in order to determine θ_1. The rotation from frame 0-to-1, θ_1, should, intuitively, equal the sum of the rotations from 0 to 5 and 5-to-1. The angle φ_1 can be found by examining the triangle with sides $0P5y$ and $0P5x$.

$$\varphi_1 = tan^{-1}(\frac{0P5y}{0P5x}) \tag{5}$$

The angle φ_2 is found by examining the rightmost triangle with φ_2 as one of the angles. Two of the sides have lengths d_4 and $|0P5xy|$:

$$\varphi_2 = \pm cos^{-1}(\frac{d_4}{|0P5xy|}) = \pm cos^{-1}(\frac{d_4}{\sqrt{(0P5y)^2 + (0P5x)^2}}) \tag{6}$$

The desired angle θ_1 can now be found simply as:

$$\theta_1 = \varphi_1 + \varphi_2 + \frac{\pi}{2} = tan^{-1}(\frac{0P5y}{0P5x}) \pm cos^{-1}(\frac{d_4}{\sqrt{(0P5y)^2 + (0P5x)^2}}) + \frac{\pi}{2} \tag{7}$$

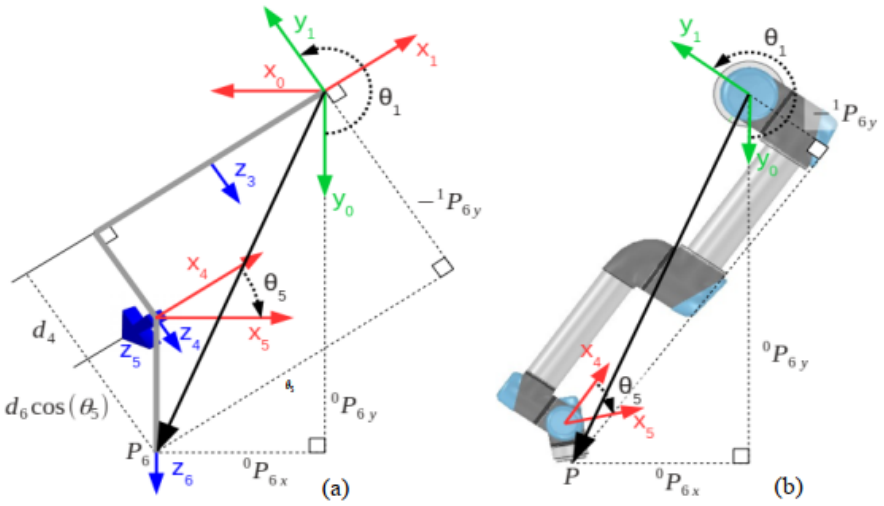


Fig. 4. UR5 robot configuration to determine θ_5

The way to figure out θ_5 now is to notice that $1P_{6y}$ is exclusively dependent on θ_5. In Fig.4, y_1 can be traced backwards to see that $1P_{6y}$ is given by:

$$1P_{6y} = -d_4 - d_6 cos(\theta_5) \tag{8}$$

The y-component of $1P_6$, $1P_{6y}$ can also be expressed by looking at $1P_6$ as a rotation of $0P_6$ around z_1.

$$1P_6 = {}^0_1R^T.0P_6 \Rightarrow \begin{bmatrix} 1P_{6x} \\ 1P_{6y} \\ 1P_{6z} \end{bmatrix} = \begin{bmatrix} c1 & s1 & 0 \\ -s1 & c1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0P_{6x} \\ 0P_{6y} \\ 0P_{6z} \end{bmatrix} \tag{9}$$

Using the second row of Eq (9),

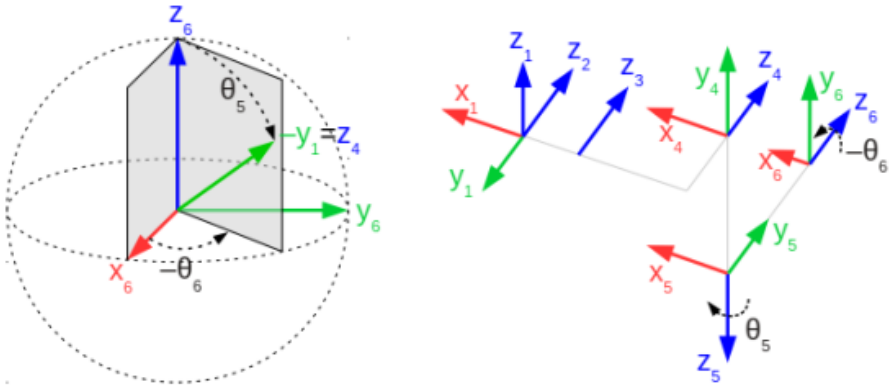
$$1P_{6y} = -(s1)0P_{6x} + (c1)0P_{6y} \tag{10}$$

Equating Eqs (8) & (10),

$$cos(\theta_5) = \frac{d_4 - s10P_{6x} + (c1)0P_{6y}}{d_6}$$

$$\theta_5 = \pm cos^{-1}(\frac{d_4 - s10P_{6x} + (c1)0P_{6y}}{d_6}) \tag{11}$$

The two solutions for θ_5 indicates to moving the wrist “down” or “up”.



(a) Polar angles for $-6Y_1$ (b) Reference view of the relevant frames

Fig. 5. Geometry used to determine azimuth angle θ_6

The axis $-6Y_1$ expressed in spherical coordinates with azimuth $-\theta_6$ and polar angle θ_5. $6Y_1$ is denoted as y_1 in the Fig.5. $-6Y_1$ is converted to Cartesian coordinates by transforming it from spherical coordinates.

$$6Y_1 = [-s5c6 \quad s5s6 \quad -c5]^T \tag{12}$$

It could be identified that $6Y_1$ is given as a rotation of θ_1 in the (x, y) plane of frame 0.

$$6Y_1 = -6X_0 sin(\theta_1) + 6Y_0 cos(\theta_1) \Rightarrow 6Y_1 = \begin{bmatrix} -6X_{0x} sin(\theta_1) + 6Y_{0x} cos(\theta_1) \\ -6X_{0y} sin(\theta_1) + 6Y_{0y} cos(\theta_1) \\ -6X_{0z} sin(\theta_1) + 6Y_{0z} cos(\theta_1) \end{bmatrix} \tag{13}$$

From Eqs (12) & (13):

$$\begin{bmatrix} -s5c6 \\ s5s6 \\ -c5 \end{bmatrix} = \begin{bmatrix} -6X_{0x} sin(\theta_1) + 6Y_{0x} cos(\theta_1) \\ -6X_{0y} sin(\theta_1) + 6Y_{0y} cos(\theta_1) \\ -6X_{0z} sin(\theta_1) + 6Y_{0z} cos(\theta_1) \end{bmatrix} \tag{14}$$

Using the first and the second rows of Eq (14):

$$c6 = \frac{6X_{0x} sin(\theta_1) - 6Y_{0x} cos(\theta_1)}{s5} \tag{15}$$

$$s6 = \frac{6Y_{0y} cos(\theta_1) - 6X_{0y} sin(\theta_1)}{s5} \tag{16}$$

Hence

$$\theta_6 = tan^{-1}\frac{6Y_{0y}cos(\theta_1) - 6X_{0y}sin(\theta_1)}{6X_{0x}sin(\theta_1) - 6Y_{0x}cos(\theta_1)} \tag{17}$$

If the denominator s5 = 0, then θ_6 is undetermined. As a result, as illustrated in Fig.5. (b), the joint axes 2, 3, 4, and 6 are aligned. Because the axes 2, 3, and 4 can rotate the end-effector around Z_6 without moving it, the 6^{th} joint is no longer needed. In this situation, θ_6 can be set to any number. The remaining three joints (2, 3, and 4) are now examined to find the remaining angles $\theta_2, \theta_3, \theta_4$.. Their joint axes are parallel.

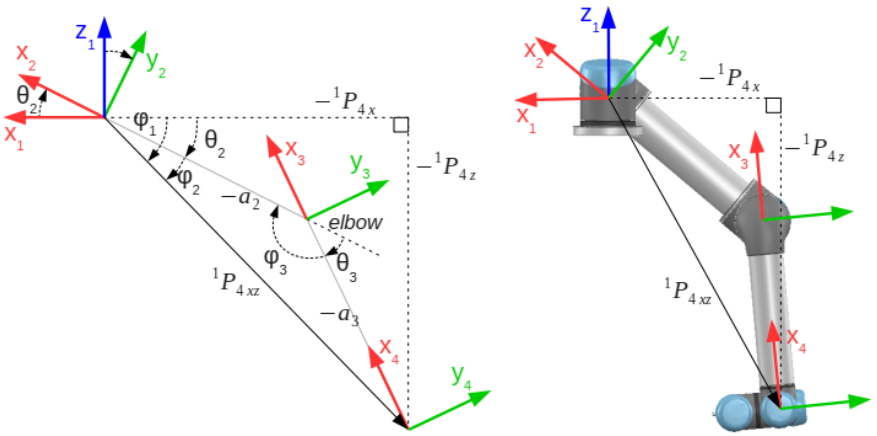


Fig. 6. Joint 2, 3, and 4 together constitutes a 3R planar manipulator

As ${}^0_1T$, ${}^4_5T$,, and ${}^5_6T$ are all known at this point, ${}^1_4T$ can be determined easily. The above picture shows this transformation in the x; z-plane of frame 1. Using the cosine law;

$$|1P_{4xz}|^2 = (-a2)^2 + (-a3)^2 - 2(-a2)(-a3)cos(\varphi_3) \tag{18}$$

$$\varphi_3 = \pm cos^{-1}(\frac{(a2)^2 + (a3)^2 - |1P_{4xz}|^2}{2(a2)(a3)}) \tag{19}$$

From the diagram shown in Fig.6 above,

$$\theta_3 = \pi - \varphi_3 = \pi \pm cos^{-1}(\frac{(a2)^2 + (a3)^2 - |1P_{4xz}|^2}{2(a2)(a3)}) \tag{20}$$

The two answers refer to "elbow move up" and "elbow move down," respectively. When the distance to the 4^{th} joint is greater than $|a2 + a3|$ or less than $|a2 - a3|$, there are no solutions. When $a2 = a3$, there is a singularity, when $\theta_3 = \pi$, making θ_2 arbitrary. θ_2 can be found by inspecting the Fig.6 above:

$$\varphi_1 = tan^{-1}(-\frac{1P_{4z}}{1P4x}) \tag{21}$$

And also using sine law,

$$\varphi_2 = sin^{-1}(\frac{-a3sin(\varphi_3}{|1P_{4xz}|}) \tag{22}$$

And $\sin(\varphi_3) = \sin(\pi - \theta_3) = \sin(\theta_3)$
Then, θ_2 can be expressed as;

$$\theta_2 = \varphi_1 - \varphi_2 = tan^{-1}(-\frac{1P_{4z}}{1P_{4x}}) - sin^{-1}(\frac{-a3sin(\theta_3)}{|1P_{4xz}|}) \tag{23}$$

The angle θ_4 is the angle from X_3 to X_4 measured about Z_4.
From the transformation matrix, ${}^3_4T$, $3X_4$:

$$\theta_4 = tan^{-1}(\frac{3X_{4y}}{3X_{4x}}) \tag{24}$$

3.2.4. *Dynamic Modelling of the UR5*

The dynamical model of UR5 is represented by using the Euler-Lagrange method. The difference between the manipulator's kinetic energy K and the potential energy P is the Euler-Lagrange. The Lagrangian L can be given by:

$$L = K - P \tag{25}$$

Where, the kinetic K and the potential energy P of the UR5 robot are:

$$K = \frac{1}{2}\sum_{i=1}^{6}(m_i v_i^T v_i + \dot{\theta}_i^T I_i \dot{\theta}) \tag{26}$$

$$P = \sum_{i=1}^{6} g^T r_{ci} m_i \tag{27}$$

Where, m_i is the mass of link i, inertial matrix is I_i calculated in a coordinate frame parallel to frame i with its origin at the center of mass, v_i is the linear velocity and $\dot{\theta}_i$ is the angular velocity, g is inertial frame vector in the direction of gravity and the vector r_{ci} gives the center of mass of link i coordinates.

Having the Lagrangian of the UR5 is, equations of motion can be determined using the Euler-Lagrange equation, which is shown as follow:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial(\dot{\theta})_i}\right) - \frac{\partial L}{\partial \theta_i} = \tau_i \tag{28}$$

Where, τ_i is the sum of external torques for a rotational motion and i=1, 2...6, $\dot{\theta}_i = \frac{\partial \theta_i}{\partial t}$.

According to Lagrangian method, the following equation can be used to determine the dynamic model of the UR5 robot.

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) + f_v\dot{\theta} + F_c sign(\dot{\theta}) = B\tau_m \tag{29}$$

Where,$\theta = [\theta_1, \theta_2, \ldots, \theta_6]^T$ are independent joint coordinates vector, $M(\theta)$is the symmetric positive definite inertial mass matrix of the system, $\mathrm{C}\left(\theta,\dot{\theta}\right)$ is the Coriolis and centrifugal terms matrix, $\mathrm{G}(\theta)$ is the gravity terms vector while τ is the control vector. Vector τ_{m} includes motor torques, and B indicates distribution force matrix. The vector of Coulomb friction forces Fc, as well as viscous joint friction coefficients diagonal matrix f_v [15].

3.2.5. *UR5 Trajectory Planning*

Consider one of the joints that is θ_i at the start of the motion segment at time t_i and that want to change to a new value of θ_f at time t_f. One method is to design the trajectory with a polynomial so that the initial and final boundary conditions match what is already known, specifically that θ_i and θ_f are known and the velocities and the accelerations at the start and end of the motion segment are zero or other known values. The fifth order trajectory can be given by:

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \tag{30}$$

$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 \tag{31}$$

$$\ddot{\theta}(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3 \tag{32}$$

Where, c_0, c_1, c_2, c_3, c_4, and c_5 are constants that has to be determined based on the boundary conditions, $\theta(t)$, $\dot{\theta}(t)$, and $\ddot{\theta}(t)$ are the angular position, velocity and acceleration of the joint respectively.

3.3. *Data collection*

The dataset consisted of various objects including toy blocks of different shapes and colors, as well as novel objects like bottles, bananas, screwdrivers, cups, forks, and adjustable wrenches. The objects were arranged in three different configurations for testing:

- Randomly arranged clutter.
- Challenging well-ordered configurations.
- Novel object configurations.

The state s_t of the proposed system is represented as an RGB-D heightmap image of the scene at time t. The RGB-D images from a fixed-mount RGB-D camera is taken, project the data into a 3D point cloud, then back-project orthographically upwards in the gravity direction using a known extrinsic camera parameter to calculate both color (RGB) and height-from-bottom (D) channels heightmap images. The RGB-D camera's additional depth information is critical for robots that interact with a three-dimensional environment. The working environment area covered a 448 by 448 mm tabletop surface.

Working with raw heightmap pictures, which are 224 by 224 pixel RGB-D images, can be time- consuming and memory-intensive [16]. Normalization of input data is a critical step that assures that each input heightmap has a consistent data distribution. Convergence of the deep neural network is accelerated as a result of this. heightmap normalization in a heightmap image is accomplished by subtracting the mean from each pixel and dividing the result by the standard deviation.

The input heightmap is rotated by 36 orientations θ, each of which corresponds to a push or grasp action with distinct products of 10° angle from the original state, before being sent to the densnet-121 to generate a set of 36 pixelwise Q-value maps to make learning-oriented pushing and grasping actions easier.

3.4. *Deep Reinforcement Learning*

In deep reinforcement learning, the UR5 uses a deep neural network to learn complex features and decides on its own what action to take by interacting with its environment. The UR5 strives to discover the best decision-making method that will allow it to maximize the rewards acquired over time through its experience.

As it allows the interaction process of reinforcement learning to be defined in probabilistic terms, the Markov Decision Process serves as the theoretical foundation for reinforcement learning. Representing all possible states that the UR5 could find itself in by the set $S = \{s_1,\ s_2,\ s_3,\ \ldots,\ s_n\}$ and the set of actions that the UR5 can execute to the system to translate from on state to the next state by $A = \{a_1,\ a_2,\ a_3,\ ...,\ a_m\ \}$ is a common practice. At a given time t the UR5 chooses and performs an action a_t based on the optimal policy, $\pi^*(s_t)$ and will cause the state to change from s_t to s_{t+1}. In response to this transition of state the environment gives a reward r_t to the UR5. After receiving the rewardr_t, the

UR5 is in state $s_{t+1}$and performs the next actiona_{t+1}. This action will cause the environment to change to state s_{t+2}. And that will be followed by the environment granting the robot a reward r_{t+1}, this process is repeated for each time step.

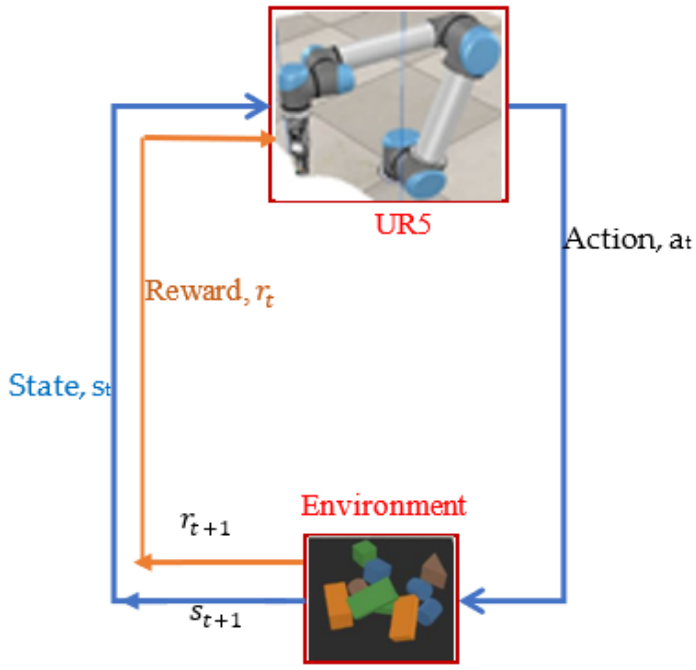


Fig. 7. Deep Reinforcement Learning (DRL) UR5-Environment interaction

The deep reinforcement learning problem is used to find the best policy $(\pi^* : S \rightarrow A)$ that maximizes the expected sum of future reward$R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ for performing an action over time horizon T and the discount factor$\gamma \in [0, 1]$ which accounts for degree of the importance of the future rewards at the present state. Deep Q-learning was used to train an optimal policy $\pi^*(s_t)$ that choses action from $A = \{a_1, a_2, a_3, ..., a_m\}$ that maximizes a state-action value function $Q_\pi(s_t, a_t)$, a measure of the reward for performing action a_t in state s_t at time t.

The equation to determine the new update Q-value for state-action pair (s, a) at time t is given by the Bellman equation;

$$Q(s_t, a_t)^{New} \leftarrow Q(s_t, a_t) + \alpha (r_t + \gamma argmax_a Q(s_{t+1}, a) - Q(s_t, a_t)) \tag{33}$$

where, a is a set of all future available actions, α is the learning rate $(0 < \alpha \leq 1)$, $Q(s_t, a_t)$ is the current Q-value that has to be updated, r_t is an immediate reward, $argmaxQ(s_{t+1}, a)$ is an estimation of the optimal future reward value.

The two important aspects of the DQN algorithm, as proposed by [16], are the use of a target network and experience replay. In this work, the target network φ', which is parameterized by θ^- and is the same with the normal DQN except the parameters are updated every C time steps from the DQN, was used. So that at every C step, $\theta = \theta^-$ and kept fixed on all other steps. The target used by DQN at the i^{th} layer, is then:

$$y_i = r_t + \gamma max_a Q_{\theta^-}(s_{t+1}, a) \tag{34}$$

Minimizing the temporal difference error δ_t of $Q_\theta(s_t, a_t)$ to a fixed target value between individual updates, y_i is the objective of the learning. Therefore, the temporal difference δ_t can be given by;

$$\delta_t = |r_t + \gamma max_a Q_{\theta^-}(s_{t+1}, a) - Q_\theta(s_t, a_t)| \tag{35}$$

The main idea of the experience replay mechanism is that during the DQN training process, the UR5 saves every experience tuple e_t at each time point$t$, $e_t = \{s_t\ ,\ a_t\ ,\ r_t\ ,\ s_{t+1}\ \}$. The experience e_t would be used as training data to update the DQN weights and the biases. The experience tuple is stored in the replay memory M of length N, where $M = \{e_1,\ e_2\ ,\ e_3\ , \ldots ,\ e_N\ \}$. A stochastic rank-based prioritization for prioritizing experience was employed in this study so that significant transitions might be replayed more frequently and so learned more effectively. In particular, the probability of sampling transition i from the replay buffer of size N is defined as;

$$p(i) = \frac{{D_i}^\alpha}{\sum_{k=0}^{N} {D_k}^\alpha} \tag{36}$$

where $D_i > 0$ denotes the priority of transition i, and the exponent α denotes the amount of prioritizing employed. As a result, this value determines how much prioritization is employed. In particular, the transitions i in the replay buffer is ranked by the absolute value of temporal difference (TD) error.

$$D_i = 1/rank(i) \tag{37}$$

where, $rank(i)$ is the rank of the transition i in the replay buffer sorted according to the temporal difference error δ_t.

3.4.1. *State representation*

The state s_t of the proposed system is represented as an RGB-D heightmap image of the scene at time t. The RGB-D images from a fixed-mount RGB-D camera is taken, project the data into a 3D point cloud, then back-project orthographically upwards in the gravity direction using a known extrinsic camera parameter to calculate both

color (RGB) and height-from-bottom (D) channels heightmap images. The RGB-D camera's additional depth information is critical for robots that interact with a three-dimensional environment. The working environment area covered a 448 by 448 mm tabletop surface.

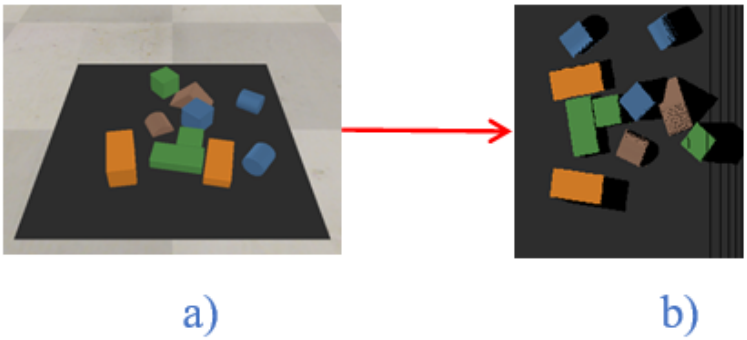


Fig. 8. (a) RGB color image and (b) RGB heightmap

Working with raw heightmap pictures, which are 224 by 224 pixel RGB-D images, can be time- consuming and memory-intensive [16]. Normalization of input data is a critical step that assures that each input heightmap has a consistent data distribution. Convergence of the deep neural network is accelerated as a result of this. heightmap normalization in a heightmap image is accomplished by subtracting the mean from each pixel and dividing the result by the standard deviation.

The input heightmap is rotated by 36 orientations θ, each of which corresponds to a push or grasp action with distinct products of 10 ° angle from the original state, before being sent to the densnet-121 to generate a set of 36 pixelwise Q-value maps to make learning-oriented pushing and grasping actions easier.

3.4.2. *Densely Connected Convolutional Neural Networks*

The deep Q-function was modelled as two feed-forward fully connected convolutional networks (FCNs) for each motion primitive behavior; one for pushing network, $\emptyset_p$ and another for grasping networks, $\emptyset_g$ by extending the vanilla deep Q-networks (DQN) [16]. It has recently been demonstrated that to train deeper, more accurate and more efficient features, the connection between layers next to the input and close to the output of the convolutional network should be shorter [17]. The normalized and rotated heightmap is fed to the densenet-121 network.

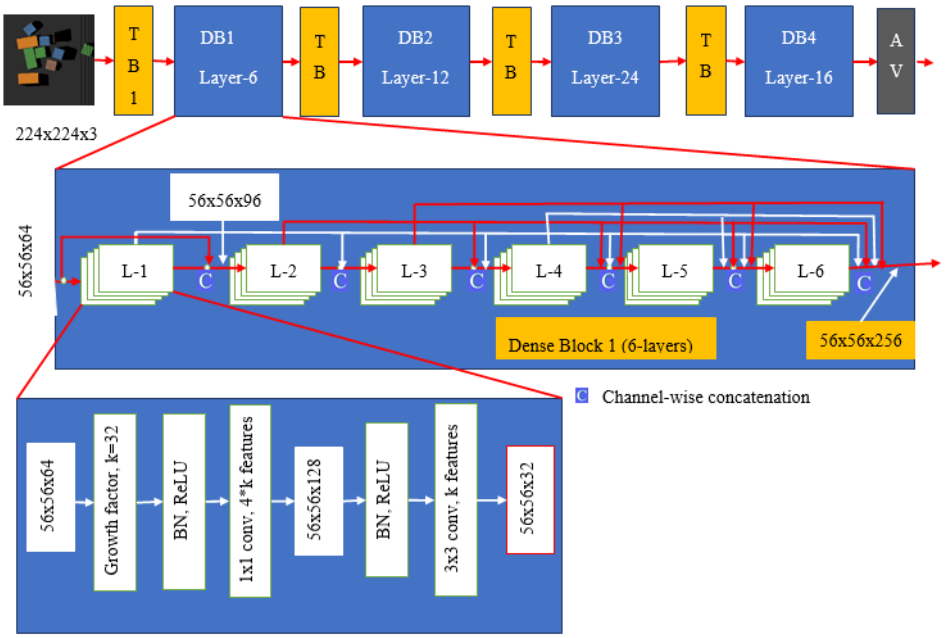


Fig. 9. A deep DenseNet-121 with four dense blocks

The DenseNet-121 is used in both the push and grasp neural networks. The depth and color heightmaps are concatenated in channel-wise order after the DenseNet-121 network, followed by two additional similar blocks with batch normalization, nonlinear activation function (ReLU), and convolutional layers for feature embedding. The pixel-wise state-action prediction value $Q(s_t, a_t)$ then bilinearly up sampled. The Q-value guess at pixel p depicts the expected reward of performing primitive action at $3D$ position $q(x, y, z, \theta)$ where q maps from pixel $p \in s_t$, and bilinear upsampling creates the same heightmap size and resolution as s_t of each dense 36 pixel-wise map of Q-values [18].

3.4.3. *Pushing Primitive Actions*

The beginning points of a 10 cm push and the push direction in one of k=36 orientations are denoted by the push primitive actionψ_p, $q(x, y, z, \theta)$. The goal of pushing primitive action is to learn pushes so that subsequent grasps can be made with as few steps as feasible. Among all possible Q-values of the push Net $\emptyset_p$, an action with maximum Q-value is selected as a push action. The tip of a closed two-finger gripper is used to physically execute the push action in this thesis.

$$push\ action, a_{push} = arg\max Q_{push}\,(s_t, a_t\ ) \tag{38}$$

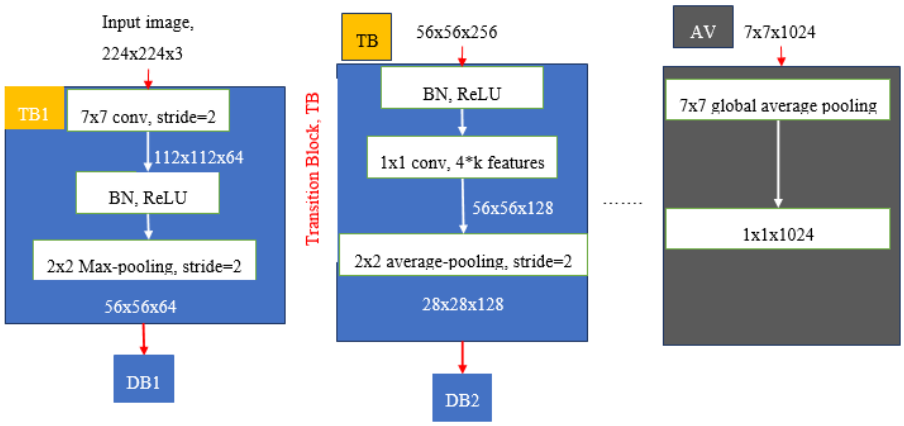


Fig. 10. DenseNet-121 Layers

Layers	*Output layer*	*DenseNet-121*
Convolution	*112x112x64*	*BN, ReLU, 7x7 conv with stride=2, 3x3 Zero padding*
Pooling	*56x56x64*	*3x3 max-pooling with stride=2*
Dense Block 1 (DB1) (6-layers)	*56x56x256*	*BN, ReLU,* $[1x1\ conv,\ 4*k\ fatures]\,x6$*, BN, ReLU,* $[3x3\ conv,\ k\ fatures]\,x6$
Transition Layer 1	*28x28x128*	*BN, ReLU, 1x1 conv with* $4*k\ fatures$*, 2x2 average pooling with stride=2*
Dense Block 2 (DB2) (12-layers)	*28x28x512*	*BN, ReLU,* $[1x1\ conv,\ 4*k\ fatures]\,x12$ *, BN, ReLU,* $[3x3\ conv,\ k\ fatures]\,x12$
Transition Layer 2	*14x14x256*	*BN, ReLU, 1x1 conv with* $8*k\ fatures$*, 2x2 average pooling with stride=2*
Dense Block 3 (DB3) (24-layers)	*14x14x1024*	*BN, ReLU,* $[1x1\ conv,\ 4*k\ fatures]\,x24$*, BN, ReLU,* $[3x3\ conv,\ k\ fatures]\,x24$
Transition Layer 3	*7x7x512*	*BN, ReLU, 1x1 conv with* $16*k\ fatures$*, 2x2 average pooling with stride=2*
Dense Block 4 (DB4) (16-layers)	*7x7x1024*	*BN, ReLU,* $[1x1\ conv,\ 4*k\ fatures]\,x16$*, BN, ReLU,* $[3x3\ conv,\ k\ fatures]\,x16$
Output Layer	*1x1x1024*	*7x7 global average pooling*

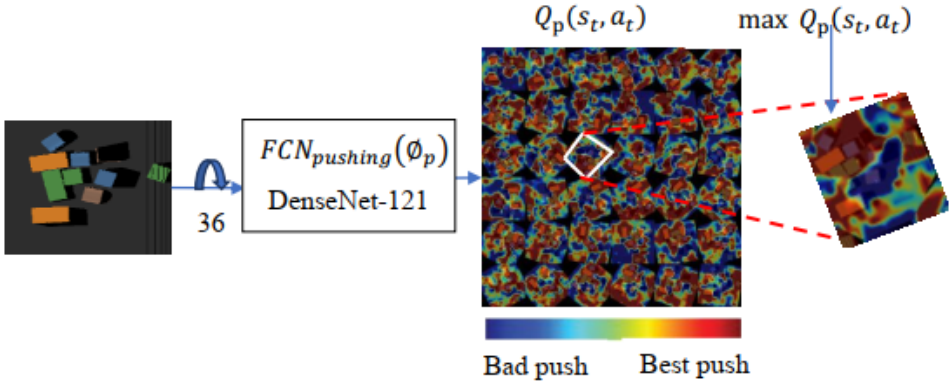


Fig. 11. Push Net ($\emptyset_p$) Q-values

3.4.4. *Grasping Primitive Actions*

Similarly, among all possible Q-values of the grasp Net $\emptyset_g$, a grasp action with maximum Q- value is selected as best grasp action.

$$grasp\ action, a_{grasp} = arg\max Q_{grasp}\left(s_t, a_t\ \right) \tag{39}$$

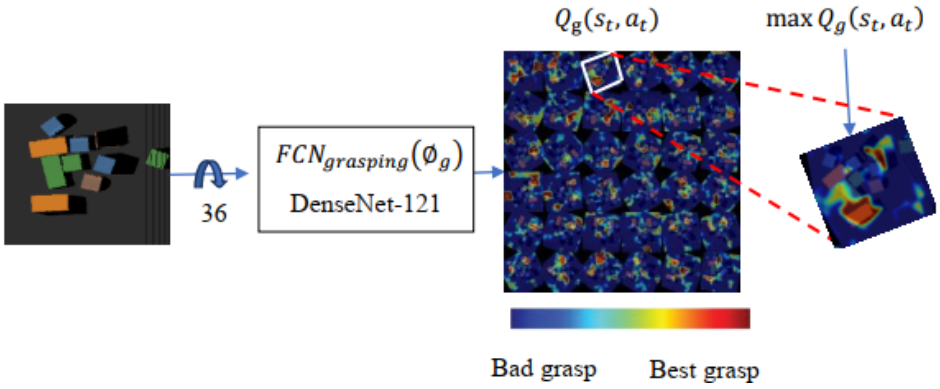


Fig. 12. Grasp Net ($\emptyset_g$) Q-values

The predicted Q-values are represented as a heatmap, with hotter locations indicating higher Q-values. White cycle in the heightmap image highlights the highest Q-value map, which correlates to the best grip action. $q(x, y, z, \theta)$ signifies the median position of a top-down parallel-jaw grab in one of k = 36 orientations in grasping primitive action. Both fingers attempt to move 3cm below $q(x, y, z)$ during a grip attempt before closing the fingers at the object's point of grasp.
Action selection strategy is based on the maximum of the primitive motions (FCNs) $\emptyset_g$ and $\emptyset_p$. The primitive action in the proposed model is the maximum of Q-values prediction of the best push action and best grasp action Q-value predictions.

$$a_t = argmax\left[\max Q_{\text{pushing}}\left(s_t, a_t\right), \max Q_{grasping}\left(s_t, a_t\right)\ \right] \tag{40}$$

At each iteration of the training session, the Huber loss function to train Q-learning on FCNs is employed [13]. For the i^{th} iteration, the Huber lose function can be written as follows:

$$H_{(lose_i)} = \begin{cases} \frac{1}{2}\left(Q^{\theta_i}\left(s_i, a_i\right) - {y_i}^{\theta_i^-}\right)^2 & if\ \left|Q^{\theta_i}\left(s_i, a_i\right) - {y_i}^{\theta_i^-}\right| < 1, \\ \left|Q^{\theta_i}\left(s_i, a_i\right) - {y_i}^{\theta_i^-}\right| - \frac{1}{2} & otherwise, \end{cases} \tag{41}$$

where, $y_i = r_t + \gamma max_a Q(s_{t+1}, a)$, θ_i denotes the parameters (weights and biases) of the neural network at the i^{th} iteration, θ_i^- signifies the target network parameters between individual updates.

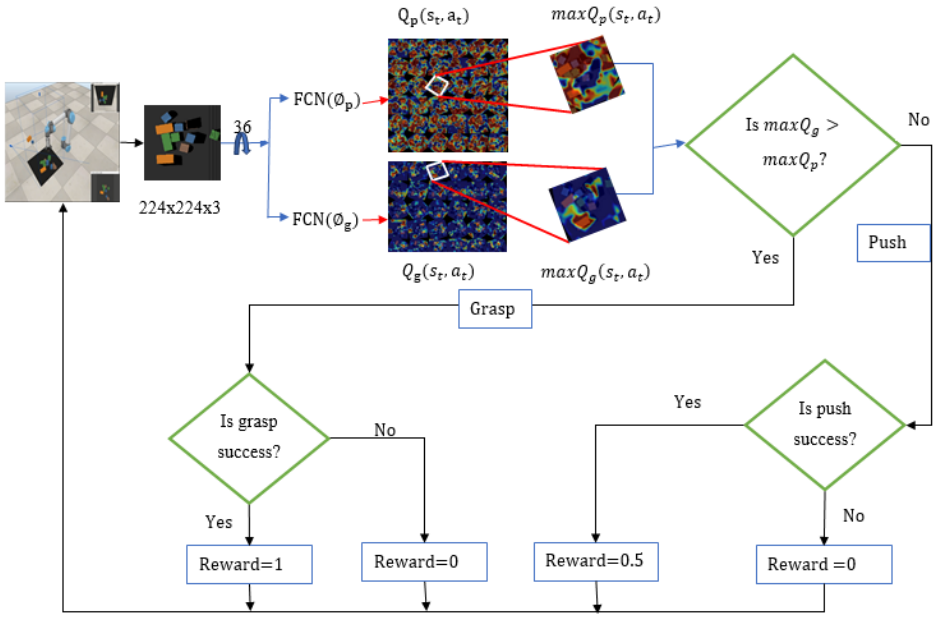


Fig. 13. Framework of pushing and grasping objects

The hyperparameters are tuned by an extensive technical trial and error experimentation in simulation and the optimal learning rate (α) of 10^{-4}, a momentum coefficient (β) of 0.9, and a weight decay factor (λ) of 2^{-5} to train fully connected convolutional neural networks are selected. During the training session, the robot learned exploration using the epsilon greedy (ϵ-greedy) policy, which started at 0.5 and was decreased to 0.1 via exponential decay. The robot begins by exploring more, and as it gains more experience via trial and error, it begins to exploit more what it has learned. And future reward discount factor was set to a fixed amount of 0.5. Over the training iterations, the gripping action's learning performance improved incrementally.

3.4.5. *Reward modeling*

A successful grasp is assigned a reward value as shown below:

$$R_g(s_t,\ s_{t+1}) = 1 \tag{42}$$

Grasp success is confirmed when the antipodal distance between gripper fingers after a grasp attempt is greater than the threshold value of the antipodal distance. While for a successful push, the reward value is modeled as:

$$R_p\left(s_t,\ s_{t+1}\right) = 0.5 \tag{43}$$

If there is a detectable change in the environment, the push is successful. The pixel-wise depth heightmap image differencing approach subtracts data from the previous depth heightmap image from those of the current depth heightmap image, to calculate the change sensed in the. Pixel-wise depth heightmap image differencing (PDD) is calculated by subtracting the current state's depth heightmap image from the previous state's depth heightmap image [18]. That is:

$$PDD = \sum |s_t - s_{t-1}| \tag{44}$$

The robot's workspace was examined with various threshold values and discovered that this may be determined by trying different experiments. Increasing the threshold value implies that the robot can detect if there is only a considerable change. When the threshold value is reduced, the robot can detect even minor changes that the human eye might miss. If the pixel-wise depth heightmap image differencing (PDD) exceeds the threshold value, a change observation is evaluated with the acquired threshold value τ:

$$change\ detected = \sum |s_t - s_{t-1}| > \tau \tag{45}$$

In general, if the PDD value is greater than the threshold value (PDD $> \tau$) and the grasping attempt is successful, a change has occurred. Aside from that, there has been no change in the workspace.

3.4.6. *Training scenarios*

The proposed model experiment was performed using DELL E2417H Desktop with NVIDIA Quadro P2000 Intel(R) Xeon(R) E-2124 CPU @3.30GHz with Ubuntu 18.04 LTS operating system, Python, PyTorch, OpenCV and V-REP.

VREP is a general-purpose robot simulation framework that is versatile, scalable, and powerful. Kinematics, dynamics, collision detection, motion planning, and mesh-mesh distance calculation modules are among the various calculation modules in V-REP [19]. V-REP was utilized in conjunction with Bullet 2.83 Physics, an open-source collision detection, soft body, and rigid body dynamics package [20]. It's used to identify collisions, resolve collisions, and resolve other limitations.

In this investigation, a UR5 robot with an RG2 parallel jaw gripper was used. The robot uses a stationary RGB-D camera to observe its surroundings. With a pixel resolution of 640 x 480, the camera gathers picture and depth maps.

During training, ten objects with various sizes and colors are dropped at random into a 448x448 mm robot's workspace/environment. By trial and error, the robot learned to perform either a grip or a push action based on the highest Q-value. Following the removal of all objects from the workspace, a new batch of 10 objects was dropped at random for additional training. Data was collected constantly until the robot had completed 3000 training iterations.

3.4.7. *Testing scenarios*

Randomly arranged, challenging arrangements of ordered objects and novel objects are provided throughout test scenarios. Bottles, various sized and shaped cups, a screw driver, a banana, and a scissor are among the novel objects.

To assess the performance and generality of the proposed model, the following metrices measurements were used, with the greater the value, the better.

(1) Average percent clearance: Over the n test runs, the average percent clearance rate assesses the policy's ability to complete the task by picking up all objects in the workspace without failing for more than 10 times.

(2) The ratio of successful grasps in all grasp attempts per completion, which assesses the grasping policy's accuracy, is called the average percent grasp success per clearance.

$$\text{Average \% grasp success } = \frac{N\_successful\ grasp}{Total\ number\ of\ trials} * 100\% \tag{46}$$

(3) The average grasp to push ratio is the number of successful grasps divided by the number of successful pushes in each test case's complete run tests.

$$\text{Average grasp to push ratio} = \frac{N\_grasp\ success}{N\_push\ success} * 100\% \tag{47}$$

where, N_successful grasp is number of successful grasps, $N\_push\ success$ is the number of successful pushes.

4. Result and discussion

In this section, the findings through this work including the training and the testing sessions are presented. After training the proposed model using self-supervised deep reinforcement learning, a different set of test scenarios with different clutter and novel objects are provided. Here, the performance of the proposed model in a cluttered environment, testing over challenging well-ordered objects and novel objects are presented.

4.1. *Performance in cluttered environments*

The results demonstrate the effectiveness of the proposed model in handling cluttered environments. Compared to traditional grasping-only policies, which achieved

a grasping success rate of just 60%, the proposed model attained a significantly higher success rate of 87%. This improvement is primarily due to the synergy between pushing and grasping actions. Pushing helps to create space around the target object, allowing the robotic gripper to secure a better grasp. The use of rewards for successful pushes further encourages this behavior, leading to a more efficient and effective grasping strategy.

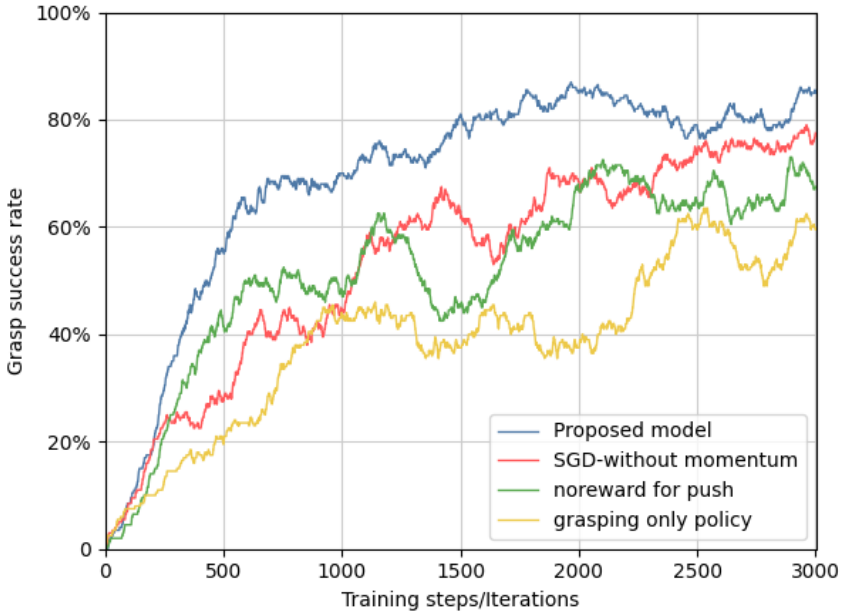


Fig. 14. Grasping performance comparison of grasping only policy, no-reward for push, SGD-without momentum and the proposed model

The above graph illustrates the grasping success rate of the grasping only policy, no-reward for push, SGD-without momentum, and the proposed model. A lot of researchers used the grasping only policy in clutter to grasp an object. In a cluttered environment, the grasping success rate of the grasping only policy is unsatisfactory. There should be room for the robotic gripper to successfully grasp an object in the clutter. Also, there are researches that trained both pushing policy and grasping policy but without reward for successful pushing policy. The grasping success rate of the proposed model has shown an interesting performance.

The proposed model reached 80% grasping success rate at 1500 iterations and the overall grasping success rate is 87%. This shows that in a clutter, synergizing pushing policies with grasping improves the grasping performance. The pushing

action is used to separate the clutter and to find room for the gripper. The purpose of pushing action is to improve future grasping. Providing a reward for successful pushes also improves a grasp success. The momentum in stochastic gradient descent is used to create an inertia to accelerate the training in the direction of minimum gradient. Momentum is used to increase the convergence rate of the training session. As shown in Fig.14, the grasping only policy grasping success rate is low. In clutter, it is recommended to use both pushing and grasping policies.

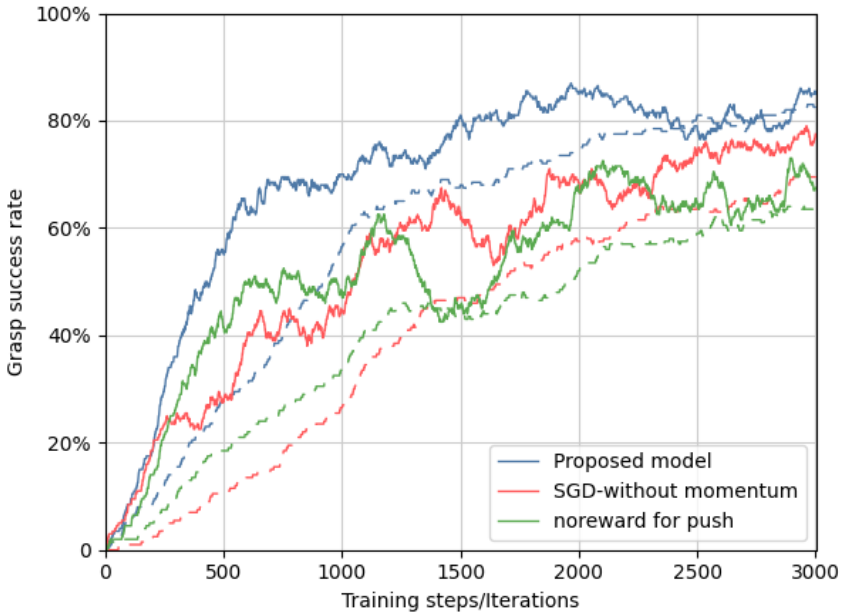


Fig. 15. Grasping success rate and push then grasp success comparison between the proposed model and SGD-without momentum

Here, the grasping performance and push then grasp success (dotted line) of the proposed model, SGD-without momentum, and no-reward for pushing action are presented. Over 3000 iterations of training, as it can be inferred from the above graph, the proposed model performed best in both grasping success rate and push then grasp success over the SGD-without momentum and no-reward for pushing actions. In the graph, the dotted lines infer the grasping successes after successful pushing actions. The grasping success rate of the proposed model has shown fast convergence. As it can be inferred from Fig. 15 over 3000 iterations, the grasping success rate of the proposed model, SGD-without momentum and no-reward for pushing action is 87%, 79%, and 71% respectively.

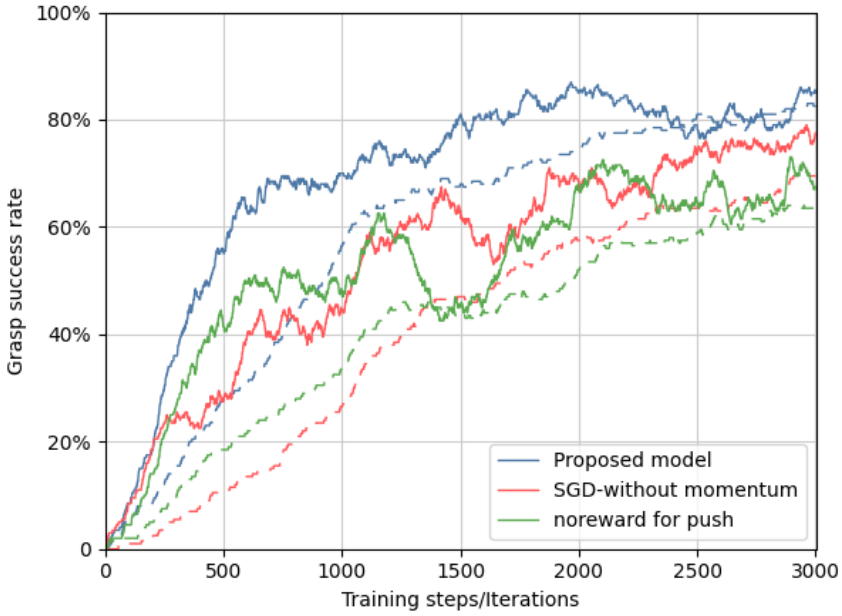


Fig. 16. Grasping success rate and push then grasp success comparison between the proposed model and SGD-without momentum

Here, the grasping performance and push then grasp success (dotted line) of the proposed model, SGD-without momentum, and no-reward for pushing action are presented. Over 3000 iterations of training, as it can be inferred from the above graph, the proposed model performed best in both grasping success rate and push then grasp success over the SGD-without momentum and no-reward for pushing actions. In the graph, the dotted lines infer the grasping successes after successful pushing actions. The grasping success rate of the proposed model has shown fast convergence. As it can be inferred from Fig.15 over 3000 iterations, the grasping success rate of the proposed model, SGD-without momentum and no-reward for pushing action is 87%, 79%, and 71% respectively.

4.2. *Randomly arranged objects*

Increasing the number of objects from 10 objects to 26 objects during training is to create more dense clutter. The model's performance was evaluated through various test scenarios, including random arrangements of objects, well-ordered challenging configurations, and novel objects. In randomly arranged environments, the model showed a robust grasp success rate, indicating its ability to generalize well to unseen configurations. Specifically, the model completed 80 test cases with a grasp

success rate of 50.5% and a grasp-to-push ratio of 77.1%. This indicates that even in highly cluttered scenes, the model can effectively manage the environment to achieve successful grasps.

The randomly arranged more cluttered objects used during testing the generalization performance of the trained model are presented in Appendix-A: Fig.A.2.

Table 2. Simulation results for random arrangements of 26 objects

Test-cases	*Completion*	*Grasp success*	*Grasp to push ratio*
Random arrangements	*80 %*	*50.5 %*	*77.1 %*

As can be inferred from Table 3, the generalizability of the proposed trained model to a more cluttered environment is interesting. The proposed model can grasp objects in clutter by synergizing push policy with grasp policy.

4.3. *Challenging well-ordered configurations*

The model's performance was further tested with three difficult test cases featuring well-ordered configurations of objects. These test cases, designed to simulate challenging gripping conditions, included objects stacked closely together and placed in orientations that complicate grasping. Despite these difficulties, the proposed model performed admirably. Test-case 00 showed a completion rate of 96.8%, with a grasp success rate of 84.7% and a grasp-to-push ratio of 99.5%. Test-cases 01 and 02 also exhibited strong performance, with grasp success rates of 61.4% and 75.1%, respectively.

Three difficult test cases with difficult clutter are provided. These configurations are created by hand to simulate difficult gripping conditions and are not used in the training process. Objects are stacked so close together in several of these test cases, in such locations and orientations. A single isolated object is placed in the workspace separately from the arrangement as a sanity check. Appendix-A contains the challenging well-ordered configuration test cases from Fig.A.3-Fig.A.5.

Table 3. Evaluation metrices of well-ordered arrangements test cases

Test-cases	*Completion*	*Grasp success*	*Grasp to push ratio*
Test-case 00	*96.8 %*	*84.7 %*	*99.5 %*
Test-case 01	*86.7 %*	*61.4 %*	*95.3 %*
Test-case 02	*96.8 %*	*75.1 %*	*100 %*

As shown from Table 4, the proposed model performs well for challenging configuration of objects in the robot work-space where the arrangements are not seen during the training session.

4.4. *Generalization to novel objects*

One of the most compelling aspects of the proposed model is its ability to generalize to novel objects. In tests with novel objects, which included items such as bottles, bananas, screwdrivers, and various sized cups, the model maintained high performance. The grasp success rates for the three novel test cases were 73.9%, 65.4%, and 95.8%, respectively. The completion rates and grasp-to-push ratios in these tests also remained high, indicating that the model can effectively handle objects with different shapes and sizes that were not part of the training set.

Simulation tests with novel objects that are more complex in shape than those employed during training are undertaken. As shown in Appendix-A from Fig.A.6 to Fig.A.8, three separate test cases with different novel objects are provided.

Table 4. Simulation results for novel object arrangements

Test-cases	*Completion*	*Grasp success rate*	*Grasp to push ratio*
Novel-test-case 00	*74.1 %*	*73.9 %*	*93.9 %*
Novel-test-case 01	*80 %*	*65.4 %*	*93.2 %*
Novel-test-case 02	*100 %*	*95.8 %*	*100 %*

Novel objects in the test were provided to understand the generalization of the proposed trained model. These are different objects that have never been seen during the training session. The objects include bottles, bananas, screwdriver, different sized cups, fork, adjustable wrench.

This generalization capability is crucial for real-world applications, where robots must handle a wide variety of objects. The ability to successfully grasp novel objects without additional training underscores the robustness and versatility of the proposed model.

4.4.1. *Discussion*

The proposed method demonstrates significant improvements in robotic manipulation within cluttered environments when compared to existing techniques as in Table 6. Here is a detailed discussion of the performance metrics:

(1) Grasp Success Rate:

Our proposed method achieves an 87% grasp success rate, which is on par with the best-performing existing technique from 2020 (87%) but significantly outperforms most other methods, particularly those from earlier

Table 5. Performance comparison with related works

Year	*Action*	*GPU*	*Epochs*	*Time*	*Work Area*	*Ref*	*Completion*	*Grasp success*	*Action Efficiency*
2021	*Grasping*	*Nvidia RTX 2080*	*2500*	*N/A*	*Clutter*	[8]	*53.1 %*	*62 %*	*51.4 %*
2020	*Grasping*	*N/A*	*7000*	*N/A*	*Sparse*	[9]	*100 %*	*87 %*	*95.2 %*
2020	*Pushing and grasping*	*Titan X (12GB)*	*10000*	*34 hrs*	*Clutter*	[10]	*93.7 %*	*76 %*	*84 %*
2020	*Pick and place*	*GeForce GTX-1660Ti*	*3000*	*10 hrs*	*Clutter*	[11]	*100 %*	*74.3 %*	*92 %*
2018	*Pushing and grasping*	*NVIDIA Titan X*	*2500*	*5.5 hrs*	*Clutter*	[12]	*100 %*	*65.4 %*	*59.7 %*
2022	*Pushing-grasping*	*Nvidia GTX 1080Ti*	*250*	*N/A*	*Clutter*	[13]	*100 %*	*83.5 %*	*69 %*
2024	*Proposed*	*DELL E2417H desktop*	*3000*	*5 hrs*	*Clutter*	*N/A*	*100 %*	*87 %*	*91 %*

years, such as the 62% success rate achieved in 2021 by Al-Shanoon et al. and the 65.4% from 2018 by Zeng et al.

(2) Action Efficiency:

The proposed approach shows an action efficiency of 91%, which is slightly lower than the highest efficiency of 95.2% reported in 2020 by Joshi et al. However, it surpasses many other methods, indicating a more balanced and efficient action strategy within cluttered environments. This is a notable improvement over methods such as Wu et al.'s 2023 approach, which reported an efficiency of 54.1%.

(3) Completion Rate:

The proposed model consistently achieves a 100% task completion rate, indicating robust performance in complex, cluttered scenarios. This matches the performance of several other top methods from 2020 to 2024, showing that the proposed method is reliable for practical applications in industrial settings.

(4) Hardware and Computational Efficiency:
The proposed method utilizes a DELL E2417H desktop, which is more accessible and cost-effective compared to high-end GPUs like the Nvidia RTX 2080 Ti or Titan X used in other studies. Despite this, it maintains competitive performance, highlighting its efficiency and potential for broader application.

(5) Training time and Epoch:
Our proposed model's training time takes about 5 hours for 3000 epochs. This shows that with less training time and epoch, our proposed model shows an interesting performance over other works.

5. Conclusion and future works

Grasping an object in a cluttered scene has been a subject of researchers, yet a challenging and less explored. It's tough to grasp an object in a cluttered environment without using non-prehensile motions. In this work, to improve grasping in a cluttered environment, a non-prehensile action, namely pushing was adopted. Using available resources, the proposed model has improved grasping an object in the clutter by synergizing pushing and grasping actions. The proposed model grasping success rate has achieved 87%. The proposed model has shown grasping success rate improvement of 27%, 16%, and 8% over grasping only policy, no-reward for pushing policy, and SGD-without momentum strategies respectively. So, it can be concluded that synergizing pushing and grasping policies improve the grasping performance of objects in a cluttered scene. Giving a reward for successful pushing action encourages synergizing the pushing and grasping actions for improved grasping success rate. Also, SGD with momentum optimization improves the convergence rate of training. The proposed framework has also shown a great generalization skill for randomly arranged dense clutter, challenging well-ordered and novel objects.

References

1. R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review, Robotics, vol. 10, no. 1, pp. 1–13, 2021, doi: 10.3390/robotics10010022.
2. D. S. Zamarrón, N. I. A. de las Casas, E. G. Grajeda, J. F. Alatorre Ávila, and N. A. Jorge Uday, Educational robot arm development, Comput. y Sist., vol. 24, no. 4, pp. 1387–1401, 2020, doi: 10.13053/CYS-24-4-3165.
3. G. Porcelli, Dynamic Parameters Identification of a UR5 Robot Manipulator, no. July, p. 82, 2020.
4. L. F. Huang, Artificial intelligence, vol. 4. 2010.
5. V. Sharma, S. Rai, and A. Dev, A Comprehensive Study of Artificial Neural Networks, Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 2, no. 10, pp. 278–284, 2012, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.9353&rep=rep1&type=pdf.

6. M. A. Yasin, W. A. M. Al-Ashwal, A. M. Shire, S. A. Hamzah, and K. N. Ramli, Tri-band planar inverted F-antenna (PIFA) for GSM bands and bluetooth applications, ARPN J. Eng. Appl. Sci., vol. 10, no. 19, pp. 8740–8744, 2015.
7. J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, How to train your robot with deep reinforcement learning: lessons we have learned, Int. J. Rob. Res., vol. 40, no. 4–5, pp. 698–721, 2021, doi: 10.1177/0278364920987859.
8. A. Al-Shanoon, H. Lang, Y. Wang, Y. Zhang, and W. Hong, Learn to grasp unknown objects in robotic manipulation, Intell. Serv. Robot., vol. 14, no. 4, pp. 571–582, 2021, doi: 10.1007/s11370-021-00380-9.
9. C. S. C. Marwan Qaid Mohammed, Kwek Lee Chung, Pick and Place Objects in a Cluttered Scene Using Deep Reinforcement Learning, Int. J. Mech. Mechatronics Eng. IJMME, vol. 20, no. 04, pp. 50–57, 2020.
10. S. Joshi, S. Kumra, and F. Sahin, Robotic Grasping using Deep Reinforcement Learning, IEEE Int. Conf. Autom. Sci. Eng., vol. 2020-Augus, no. July, pp. 1461–1466, 2020, doi: 10.1109/CASE48305.2020.9216986.
11. A. Hundt et al., Good Robot!': Efficient Reinforcement Learning for Multi-Step Visual Tasks with Sim to Real Transfer, IEEE Robot. Autom. Lett., vol. 5, no. 4[1] A. Hundt et al., "'Good Robot!': Efficient Reinforcement Learning for Multi-Step Visual Tasks with Sim to Real Transfer," IEEE Robot. Autom. Lett., vol. 5, number 4, pp. 6724–6731, 2020, doi: 10.1109/LRA.2020.3015448., pp. 6724–6731, 2020, doi: 10.1109/LRA.2020.3015448.
12. Y. Yang, H. Liang, and C. Choi, A Deep Learning Approach to Grasping the Invisible, IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 2232–2239, 2020, doi: 10.1109/LRA.2020.2970622.
13. A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, Learning Synergies between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning, IEEE Int. Conf. Intell. Robot. Syst., pp. 4238–4245, 2018, doi: 10.1109/IROS.2018.8593986.
14. P. M. Kebria, S. Al-Wais, H. Abdi, and S. Nahavandi, Kinematic and dynamic modelling of UR5 manipulator, 2016 IEEE Int. Conf. Syst. Man, Cybern. SMC 2016 - Conf. Proc., pp. 4229–4234, 2017, doi: 10.1109/SMC.2016.7844896.
15. P. Boscariol, R. Caracciolo, D. Richiedei, and A. Trevisani, Energy optimization of functionally redundant robots through motion design, Appl. Sci., vol. 10, no. 9, 2020, doi: 10.3390/app10093022.
16. V. Mnih et al., Human-level control through deep reinforcement learning, Nature, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.
17. G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, Densely connected convolutional networks, Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 2261–2269, 2017, doi: 10.1109/CVPR.2017.243.
18. C. S. C. Marwan Qaid Mohammed, Kwek Lee Chung, Pick and Place Ob-

jects in a Cluttered Scene Using Deep Reinforcement Learning, Int. J. Mech. Mechatronics Eng. IJMME, vol. 20, no. 04, pp. 50–57, 2020.

19. E. Rohmer, S. P. N. Singh, and M. Freese, V-REP: A versatile and scalable robot simulation framework, IEEE Int. Conf. Intell. Robot. Syst., pp. 1321–1326, 2013, doi: 10.1109/IROS.2013.6696520.
20. E. Coumans, Bullet 2 . 82 Physics SDK Manual Table of Contents, 2013.

Appendix: The proposed model training and test cases

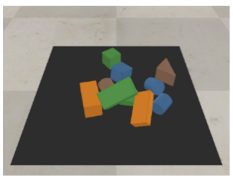


Fig.A.1. Training objects

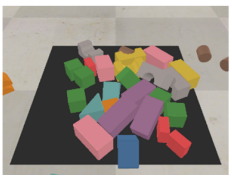


Fig.A.2. Random configuration

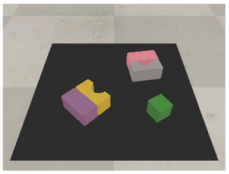


Fig.A.3. test-case-00

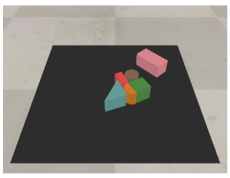


Fig.A.4. Test-case-01

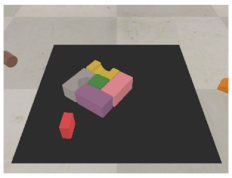


Fig.A.5. Test-case-02

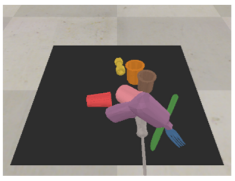


Fig.A.6. Novel-test-case-00

Fig.A.7. Novel-test-case-01

Fig.A.8. Novel-test-case-02

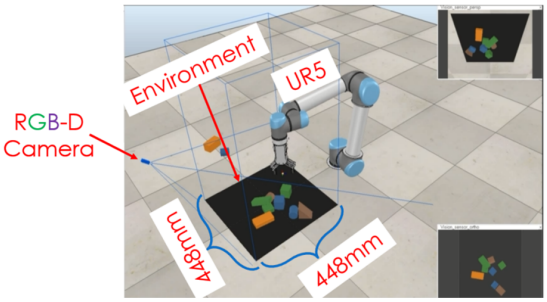


Fig.A.9 UR5-Environment interaction of the proposed model