Training an Under-actuated Gripper for Grasping Shallow Objects Using Reinforcement Learning

Wael M. Mohammed
FAST-Lab, Faculty of Engineering and Natural Sciences
Tampere University
Tampere, Finland
wael.mohammed@tuni.fi

Mirosław Nejman
Department of Automation and Metal Cutting
Warsaw University of Technology
Warsaw, Poland
miroslaw.nejman@pw.edu.pl

Fernando Castaño
Centre for Automation and Robotics
Spanish National Research Council
Technical University of Madrid
Arganda del Rey, Spain
fernando.castano@car.upm-csic.es

Jose L. Martinez Lastra
FAST-Lab, Faculty of Engineering and Natural Sciences, Tampere University
CINTECX, Universidade de Vigo
Vigo, Spain
jose.martinezlastra@tuni.fi, lastra@uvigo.es

Stanisław Strzelczak
Lab of Industrial Ecosystemics, Faculty of Production Engineering
Warsaw University of Technology
Warsaw, Poland
s.strzelczak@pw.edu.pl

Alberto Villalonga
Centre for Automation and Robotics
Spanish National Research Council
Technical University of Madrid
Arganda del Rey, Spain
alberto.villaloga@car.upm-csic.es

***Abstract*—robot programming and training depends on the task that needs to be completed, the end-effector properties and functionalities and the working space. These considerations can complicate the programming process, which in return, increase the time that is needed for training the robot. Thus, several research approaches have been introduced to address training the robots intuitively. In this regard, this paper presents an approach for training an under-actuated gripper and the robot attached to it for grasping shallow objects. The research work started by detailed analysis of the fingers of human hand during the grasping process. Then, a modified design of the gripper has been produced. This modification includes adding an artificial nail among other hardware-related modifications. Then, a Q-Learning algorithm has been used for training the gripper on grasping the shallow object. With two fingers, three actions were configured, and 625 states were configured for the learning algorithm. For the validation, a coin has been used for representing the shallow object. The results showed reduction in both the grasping time and the number of movements.**

Keywords—Reinforcement learning, Robot learning, machine learning, Grasping

I. INTRODUCTION

Robot training process has been evolving as both software and hardware technologies advance. The leap in hardware and software capabilities open the doors for new techniques and methods for programming robots in general [1], [2]. Traditionally, and still, programming robot using offline method is the most commonly used in industry. This practice provides guaranties of accuracy and assurance of the robot task execution. In this regard, several approaches have been introduced to simplify or ease the process of the robot programming [3]. As an example, the usage of force/torque sensor for online lead-through or teaching by doing or teaching be demonstration approaches. These approaches proven to be useful and sometime necessary for i.e. in medical operations or in human-robot collaboration tasks [4]. Nonetheless, the usage of them is less likely in industrial cases.

The robot motion programming depends on the end-effector and the task that the robot is required to achieve. For instance, a welding end-effector requires the consideration of the welding point when designing the motion profiles. Besides, the welding process forces the robot to adjust the path and/or the joints' angles in order to avoid possible singularities. These possible problem makes the traditional off-line programming of the robot more complicated which in return, requires more time and experience to achieve [5]. This type problems may appear in pick /grasping task as well. For i.e., the shape and the dimensions of the object play dominant role on how the robot must approach the object for grasping.

This paper aims at presenting an approach for teaching a robot to grasp a shallow object. The robot is equipped with an under-actuated gripper. This gripper was developed based on the open source Yale Open Hand Project (YOH) from the Yale GRAB Lab at Yale University [6]. The challenge to teach the robot to grasp the coin and drop it in a box without the need to use suction cups. This operation mimics the human method for grapping object by utilizing the fingertips.

The rest of the document is structured as follows: Section II presents the related research and state of the art in the field of Robotics and manipulation and reinforcement learning. Section III presents the approach of this research. Section IV provides the implementation of the presented approach. Meanwhile section V presents the results and discussion. Finally, Section V concludes the paper and provides possible future work.

II. THEORETICAL BACKGROUND

A. Robot programming

Traditionally, Robot programming requires a trained operator with experience in robotics manipulation. Moreover, the operator must have the needed knowledge about the tasks that the robot will accomplish, the nature of the process and the robot work space [5], [7]. These requirements introduce more complications sometimes. Thus, the online programming concept have been introduced to simplify and minimize the needed effort for programming the robot. As in [8], the definition of the online robot programming includes the dynamically program the robot motion while it is on run mode such as lead-through programming using force and torque sensors to convert the forces the exerted by the human on the end-effector to translate to the robot variable space.

Another approach is using teaching by demonstration. Thus, these techniques are potentially used as the robot operator does not require knowledge on the traditional programming language of the robot [9]. Nonetheless, these methods introduce concerns about the safety of the operator as the operator needs to work behind the fence with the industrial robot. Other approach tends to provide safer and more reliable methods for training the robots on achieving the required tasks. As presented in [10], the robot is trained using specification-centred generative approach. While [11] presents an approach for controlling the motion of a robot using Deep Deterministic Policy Gradient (DDPG). As showing the presented research, the used method permitted more flexible approach for programming he robot. Furthermore, [12] presented an approach for task generalization using Deep Model Fusion with the support of Multi-objective Guided Reward technique for increasing the training efficiency.

B. Reinforcement learning for robotics

Machine learning methods have demonstrated to be suitable techniques for modelling complex systems with a wide range of applications such as smart manufacturing and intelligent transportation systems [13]. Reinforcement learning (RL) is a part of machine learning that implement the maximization of cumulative reward. Usually, the reinforcement learning used in supporting task the at requires decision making. For robotics, and as presented in [14], reinforcement learning is considered as a framework that allows designing tasks and behaviours which are hard to model. According to the same source, the challenge in robotics field is the possible of dimensionality of the states and actions which is reflected as computational cost. Furthermore, [15] presents an RL techniques called Bayesian-discrimination-function-based reinforcement learning (BRL) for generating robot action. This article claims the automatic generation of the states and the actions of the robotics motion using segmented learning.

III. THE REINFORCEMENT LEARNING APPROACH FOR GRASPING SHALLOW OBJECTS

The reinforcement leaning concept in this paper includes the reward/punishment approach while the application learns about the grasping task. This means that the reinforcement learning requires a feedback system to inform the application and the running algorithm about the achievement after each epoch. Fig 1 shows the high-level architecture of the reinforcement learning system as a Cyber-Physical System (CPS) [16]. As presented in the figure, the reinforcement learning and the grasping detection system are application running on devices which can be configured and adjusted to suit the learning process.

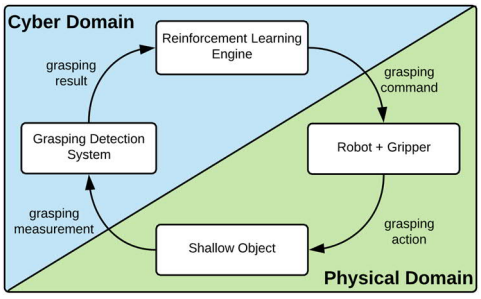

Fig 1. CPS as reinforcement learning system for grasping shallow objects

The process starts by commanding the robot to grasp the shallow object. This shallow object is a coin on a flat surface. The robot approaches the coin and try to grasp it. A detection system, like a vision system, detect if the coins is picked or not. Consequently, the detection system informs the reinforcement learning engine. Finally, the reinforcement engine updates the learning algorithm parameters by rewarding or penalizing the learning score.

For solving the problem of grasping a shallow object, a detailed observation has been conducted in order to analyse the human hand movement. As Fig 2 shows, the grasping with two fingers starting by closing the fingers until both fingers touch the coin (a, b). Then, as shown in (c, d), one nail slides under the coin while the other finger acts as support for pivoting the coin. Finally, as shown in (e, f), the needed tilt is created for grasping the coin. Therefore, the need for a gripper with 2 fingers at least is required. Additionally, these fingers have to include some sort of an edge to act as the nail on one finger and soft grippy padding to act like the other finger's end.

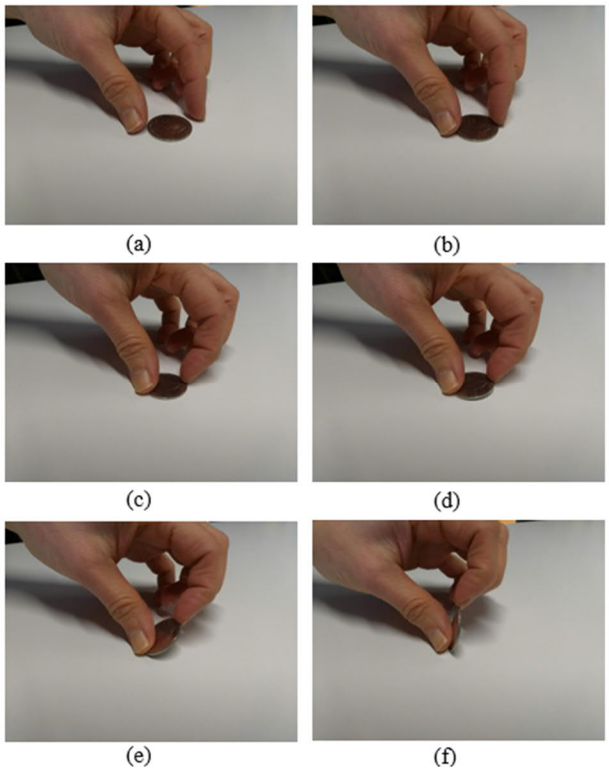

Fig 2. Steps of grasping a coin done by human

Many of the learning algorithm can be utilized for the learning task in this research. The selection was towards the Q-Learning (QL) algorithm. The QL is a reinforcement learning algorithm which does not use a model of the physical or behaviour environment of the problem. Rather, it focuses on selecting the optimal action according to the state and the score from the previous score, state and action. As equation 1 presents, the current score (Q_{i+1}) depends on the pervious score (Q_i), reward from previous action (r), learning rate (α), discount factor (γ) and possible maximum score of the current state ($maxQ_{i+1}$). In this regard, the learning rate decides the amount of the new score out of the old score. While the discount factor decides the impact of the possible current score on the current score.

$$Q_{i+1} = (1-\alpha)Q_i + \alpha(r + \gamma\, maxQ_{i+1}) \quad (1)$$

$$a_{i+1} = \begin{cases} x & ,x < \varepsilon \\ argmax(maxQ) & ,x \geq \varepsilon \end{cases} \quad (2)$$

The current action is selected based on the Finite Markov Decision Process (FMDP). As shown in equation 2, a random number (x) that is compared with the exploration rate (ε) decides the current action (a_{i+1}). If the result greater or equal than the exploration rate, then the action will be the action that scored the maximum score. Otherwise, the action will be selected randomly.

```
1: while(episode < episode_max)
2:   request coin position from camera
3:   if (coin is detected)
4:     initialization of episode parameters
5:     while(gripper is not closed OR epoch < epoch_max)
6:       select action randomly
7:       make one step in closing the gripper
7:       if (gripper is closed or epoch >= epoch_max)
8:         request coin position from camera
9:         if(coin is detected)
10:          add the penalty
11:        else
12:          add the reward
13:        end if
14:        calculate the probabilities for the next step
15:        advance epoch by 1
16:      end if
17:    end while
18:    advance episode by 1
19:  end if
20:end while
```

Fig 3. Pseudo code for the reinforcement learning algorithm

Fig 3 shows the pseudo code for the algorithm for learning grasping of shallow objects. The learning process start by looping until the application reach a predefined number of episodes. A single episode represents one trial of grapping the shallow object starting form a fully open gripper till fully closed gripper. In each episode, the application will ask for the position of the coin. The detection system, which is a smart camera that is programmed to provide the centre of a coin. In this regard, the coin is cover with white sticker in order to enhance the circle detection and therefore finding the centre accurately.

TABLE I. ACTIONS DESIGN FOR THE UNDER-ACTUATED GRIPPER

Action code	Action values	
	Finger 1 (Position counts)	*Finger 2 (Position counts)*
0	50	0
1	0	50
2	50	50

If a coin is detected, then the application will start a loop that ends if the gripper is closed or a predefined number of epochs is reached. In each epoch, the quality score is updated, and an action is selected. If the gripper is closed and the coin was picked, then the score will be awarded. Otherwise the score will be penalized. As the gripper have two under-actuated fingers, 3 actions are identified as shown in TABLE I. The actions are either finger 1 closes by 50 counts, figure 2 closes by 50 counts or both figures close by 50 counts for each one. The selection of the step value is described in the next sections.

IV. The Implementation

A. Use case description

For this research, a specific case has been built to proof the concept of robot self-learning. The case includes an ABB 140 robot with a modified YOH under-actuated gripper build in house. A National Instruments smart camera is uses for providing the feedback about the coin presence or coin location. For running the learning algorithm, a Raspberry Pi model B is used. The different component uses a local Ethernet network to communicate between each other. See Fig 4.

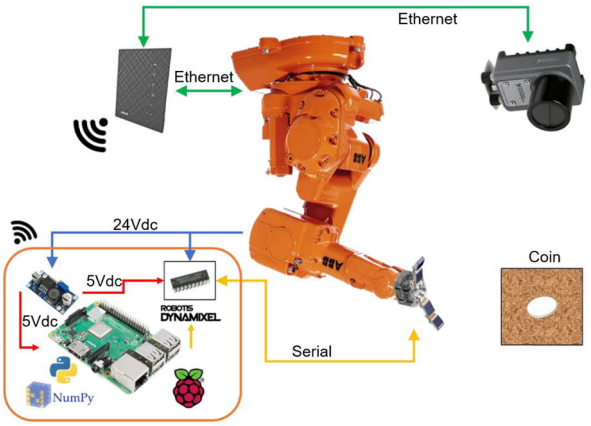

Fig 4. Use case environment

The challenge in the grasping of shallow objects is the ability to repeat the same test with a successful result. This require addressing the mechanical components as well in order to reach stable process. In this regard and based on the observation of the manual procedure depicted in Fig 2, an artificial nail (marked in red) in Fig 5 was added to one of the fingers. This addition on the under-actuated gripper forces the coin to take the same starting tilt. After that the coin start sliding between the figures until the finger close on each other.

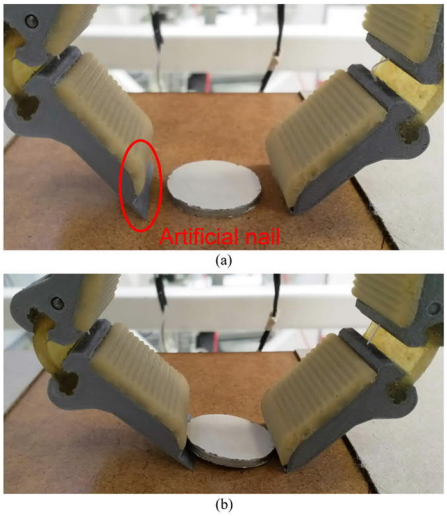

Fig 5. Under-actuated gripper with additional artificial nail. a) before applying the grasp command, b) after applying the grasp command.

It is important to mention that the gripper has been manufactured in house using 3D printing for the solid parts and moulding and then resin (PMC-780 for the flexure joints

and Vytaflex-30 for the finger pads) pouring for the flexible grippy parts as shown in Fig 6.

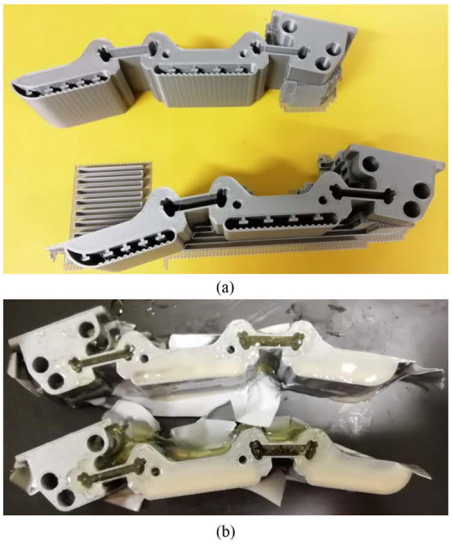

Fig 6. Fabrication of the gripper fingers. (a) 3D printed parts and (b) after pouring the risen.

For the smart camera, a simple application has been developed. The routine contains three states; *Read*, *Nothing* and *OK* as presented in Fig 7. The state *Read* occurs when the camera reading the image and then calculate the presence of the coin (coin appears as a white circle to enhance the detection). Once a coin is found, the camera goes to the *OK* state then it sends a UDP message to the Raspberry Pi informing about the coin position and diameter. If the coin was not detected, the camera sends a UDP message with all zeros as parameters then goes to the *Nothing* state. The routine then restarts after 2 seconds.

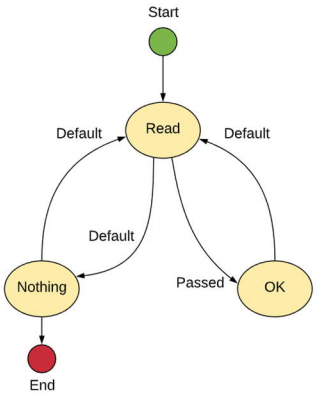

Fig 7. Camera state diagram

B. *Testing description and configuration*

For conducting the reinforcement learning, several parameters must be configured including the under-actuated gripper, the learning algorithm and the camera among other components. Starting with the fingers' actions, each of the fingers is equipped with a tendon that winds around a pulley. This pulley is connected to the servo that able to measure the rotation steps using an embedded encoder. As presented in TABLE II, each of the fingers have 1200 steps as a range between open and close. Using the trial and error, the range is divided by 50 steps (50 steps shown reasonable step for a single epoch) then the state per finger will 25 states. Thus, having 2 fingers, the total states in the system is 625 which represents the different state of the two fingers independently.

TABLE II. SETTINGS OF THE COUNTERS OF THE FINGERS

Pose	**Finger 1**		**Finger 2**	
	Position counts	***Load counts***	***Position counts***	***Load counts***
Open	2000	0	1850	0
Close	800	140	650	380

The configuration of the RL algorithm include defining the values of the awards and the penalties which are summarized in TABLE III. Using trial and error, after each action, a reward of 1 point is given. Then, if one of the fingers reaches the close condition, it will receive a penalty of 10. Following that, if the two fingers close without catching the coin, then the penalty will be 30. If the coin is successfully grasped (in the table coin is not detected), the award will be 20 if the fingers reach the close position and 25 if the fingers do not reach the complete close condition.

TABLE III. REWARD/PENALTY CONFIGURATION

Finger 1 or 2	**Coin**	**Reward**	**Description**
x	X	1	Every action returns a reward of -1
>1200	X	-10	Actions exceeding the limit have penalty of 10 and the position remains on 1200 for the finger intending to exceed 1200
1200	Yes	-30	Fingers closed and coin not extracted, gives penalty of 30
1200	No	20	Fingers closed and coin correctly extracted gives award of 20
<=1200	No	25	Coin extracted without need to close completely, gives award of 25

The last configuration in the subsections is the camera settings. Like the other settings, trial and error is used to obtain the best coin detections. As TABLE IV shows, the settings are considered to be high as the exposure is 10-13 ms. This high value selection is due to the condition of the coin as it stay stationary during the image capturing process.

TABLE IV. CAMERA CONFIGURATION

Step	**Parameter**	**Value**
Acquire image	Exposure time	10-13 ms
	Gain	100
Geometric Matching	Threshold (lower value)	200
	Minimum object size	10 mm^2
	Area filter	500 – 800 mm^2
	Minimum number of objects to pass	1

C. *Components' Interactions*

The flow diagram of the reinforcement learning interaction is illustrated in Fig 8. The activity starts once the camera sends a UDP message contains the coordinates and the circle diameter to the Raspberry Pi. Then, the RL at the Raspberry Pi calculate the position that the robot needs to move in order to grasp the coin. After moving, the robot acknowledges about the readiness for starting the training on grasping the coin. Afterwards, the RL chooses one of the actions as shown in TABLE I. This selection is done

randomly. Nonetheless, it could be selected as best previous action as the user can configure the exploration parameter. Once the action is selected, the application measures the needed steps to achieve the step. Using the serial connection, the RL that runs on the Raspberry Pi sends the command to the gripper to open one of the fingers or both fingers. This action follows the values as presented in TABLE I. Then the Raspberry Pi receives the acknowledgement about the action execution. As seen in the figure, the epoch will continue until the gripper is fully closed or the RL reaches the maximum number of the epochs.

After breaking the epochs loop, the RL command the robot to move away in order to allow the camera to find the coin. After maximum of 2 seconds, the camera sends the circle parameters. If all parameters are zeros, the gripper has successfully picked the coin and the Q score will be rewarded following TABLE III. Otherwise, it will be penalized following the same table. Finally, the Raspberry PI will command the robot to move away and the gripper to open for starting the next episode.

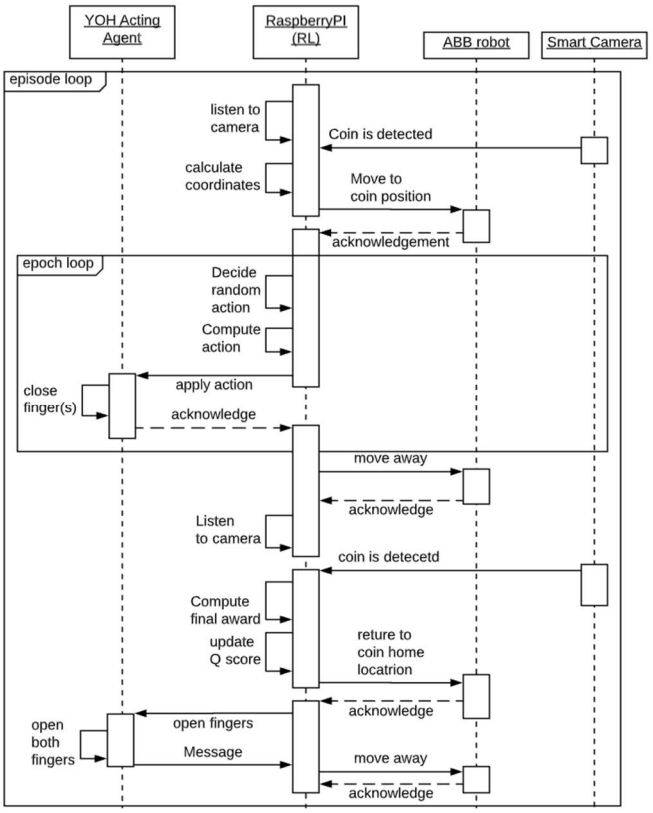

Fig 8. Sequence diagram of the reinforcement learning process

V. Results and Discussion

The tests of the previously described approach took place in the FAST-Lab (Future Automation Systems and Technologies Laboratory) premises at the Tampere University. With several attempts and fine tuning of the algorithm parameters, the selection of the epochs was 50, the episodes is 500, 0.3 for the learning rate, 0.7 for the discount factor and finally, 0.3 for the exploration rate as it is presented in TABLE V.

TABLE V. REINFORCEMENT LEARNING ALGORITHM PARAMETERS

Parameter	Value
Max. Epochs	50
Max. Episodes	500
Learning Rate (α)	0.3
Discount Factor (γ)	0.7
Exploration Rate (ε)	0.3

The training process took 3 hours and 10 minutes approximately. This process resulted in reduction of the grasping time from 24 seconds to 9 second and with reduction in movements from 2400 to 42 at show in Fig 9. The comparison has been done against force monitoring approach. The force monitoring approach has been used before for controlling the movements of the fingers. Generally, the current that is drawn by the servos represented the applied force. The monitoring was to keep the relative movement of the figures within an acceptable margin in order to keep the coin in between the figures during the grasping process

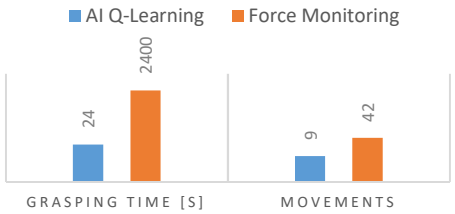

Fig 9. grasping results

The presented results show potential in using Q-Learning approach for training the gripper. However, each episode requires several minutes to be completed. This limited the possibility of having long tests that run more episodes, e.g. 5000 -7000. To overcome this, the exploration rate was increased to 0.3 where it is recommended in other research to be around 0.1. Moreover, the selection of the gripper's action was done randomly to balance the usage of both fingers. This can be more investigated as the fingers' movement is critical in this task and the success rate increase as the fingers closing synchronously as Fig 2 shows from the human behaviour. It important to mention that this training process has been done for grasping one type of shallow objects with the same dimensions. If the object will be changed, another learning process needs to be done to address the changes int eh dimensions which will affect the grasping process.

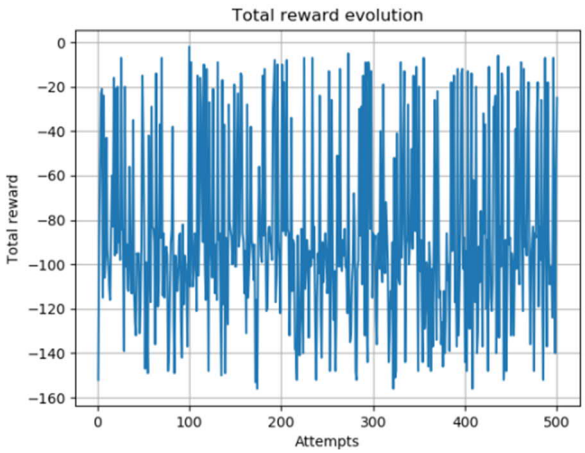

Fig 10. Evolution of the reward during the training

In addition, the reward evolution, as depicted in Fig 10, presents a random behaviour with no recognizable trend. This effect might be a result of using random selection of the actions. Finally, and as the result shows, the usage of suction cap can be faster for pick and place tasks. However, the challenge in this research to mimic the human grasping for shallow object.

VI. Conclusion

This paper presented an approach for training a robot which is equipped with an under-actuated gripper. The gripper is a modified version of the YOH open source gripper. The research started by analysing the relative motion of the human hand fingers with respect to each other during the grasping process of a coin. Afterwards, Q-Learning algorithm was selected to self-train the robot on grasping the coin by provided score after each attempt of grasping. This score is subjected to reward if the training was successful or penalty if the robot fails to grasp the coin. By having two fingers, the actions were; moving either of the fingers or both of them which resulted in having 3 actions. Meanwhile the states were 625 In order to cover all the motion of the fingers independently. The results of this research showed a potential in adapting such approach. With reduction of grasping time and number of movements, the research can be considered as a successful attempt. For the future work, the selection of the action can be changed in order to overcome the learning evolution issues. In addition, the used of different sensing method like tactile sensors or more advance vision system can be used.

Acknowledgment

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement n° 870133, correspondent to the project entitled REMODEL, Robotic Technologies for the Manipulation of Complex Deformable Linear objects

Preparation of this publication was also supported by the Polish National Agency for Academic Exchange (NAWA) through the project: "Industry 4.0 in Production and Aeronautical Engineering (IPAE)".

Finally, part of the research work in this paper has been conducted by the students Leire Amezua and Ronal Bejarano in their special assignment course.

References

[1] R. C. Luo, Y.-T. Hsu, Y.-C. Wen, and H.-J. Ye, 'Visual Image Caption Generation for Service Robotics and Industrial Applications', in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, May 2019, pp. 827–832, doi: 10.1109/ICPHYS.2019.8780171.

[2] O. De Miguel Lazaro, W. M. Mohammed, B. R. Ferrer, R. Bejarano, and J. L. Martinez Lastra, 'An Approach for adapting a Cobot Workstation to Human Operator within a Deep Learning Camera', in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finland, Jul. 2019, pp. 789–794, doi: 10.1109/INDIN41052.2019.8972238.

[3] M. Samaka, 'Implementation of AI software for the development of a robot task programming language and simulation', in *2002 IEEE International Conference on Industrial Technology, 2002. IEEE ICIT '02.*, Dec. 2002, vol. 2, pp. 869–874 vol.2, doi: 10.1109/ICIT.2002.1189282.

[4] R. Bejarano, B. R. Ferrer, W. M. Mohammed, and J. L. Martinez Lastra, 'Implementing a Human-Robot Collaborative Assembly Workstation', in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finland, Jul. 2019, pp. 557–564, doi: 10.1109/INDIN41052.2019.8972158.

[5] A. Brunete, M. Hernando, and E. Gambao, '"Hammer: Robot Programming Interface for Common People"', in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 5539–5539, doi: 10.1109/IROS.2018.8594453.

[6] 'Yale OpenHand Project - Model T42'. https://www.eng.yale.edu/grablab/openhand/model_t42.html (accessed Feb. 18, 2020).

[7] H. Friedrich, J. Holle, and R. Dillmann, 'Interactive generation of flexible robot programs', in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, May 1998, vol. 1, pp. 538–543 vol.1, doi: 10.1109/ROBOT.1998.677029.

[8] L. A. Hormaza, W. M. Mohammed, B. R. Ferrer, R. Bejarano, and J. L. Martinez Lastra, 'On-line Training and Monitoring of Robot Tasks through Virtual Reality', in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finland, Jul. 2019, pp. 841–846, doi: 10.1109/INDIN41052.2019.8971967.

[9] A. A. N. Kumaar and S. TSB, 'Mobile Robot Programming by Demonstration', in *2011 Fourth International Conference on Emerging Trends in Engineering Technology*, Nov. 2011, pp. 206–209, doi: 10.1109/ICETET.2011.30.

[10] A. Bredenfeld and G. Indiveri, 'Robot behavior engineering using DD-Designer', in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, May 2001, vol. 1, pp. 205–210 vol.1, doi: 10.1109/ROBOT.2001.932554.

[11] Z. Dwiel, M. Candadai, and M. Phielipp, 'On Training Flexible Robots using Deep Reinforcement Learning', in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 4666–4671, doi: 10.1109/IROS40897.2019.8968516.

[12] T. Wang *et al.*, 'Efficient Robotic Task Generalization Using Deep Model Fusion Reinforcement Learning', in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2019, pp. 148–153, doi: 10.1109/ROBIO49542.2019.8961391.

[13] I. L. Fe, G. Beruvides, R. Quiza, R. Haber, and M. Rivas, 'Automatic selection of optimal parameters based on simple soft computing methods. A case study on micro-milling processes', *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018, doi: 10.1109/TII.2018.2816971.

[14] J. Kober, J. A. Bagnell, and J. Peters, 'Reinforcement learning in robotics: A survey', *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013, doi: 10.1177/0278364913495721.

[15] T. Yasuda and K. Ohkura, 'A Reinforcement Learning Technique with an Adaptive Action Generator for a Multi-robot System', in *From Animals to Animats 10*, Berlin, Heidelberg, 2008, pp. 250–259, doi: 10.1007/978-3-540-69134-1_25.

[16] S. Iarovyi, W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra, 'Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems', *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1142–1154, May 2016, doi: 10.1109/JPROC.2015.2509498.