Learning Continuous Control Actions for Robotic Grasping with Reinforcement Learning

1st Asad Ali Shahid
Department of Mechanical Engineering
Politecnico di Milano
Milano, Italy
email address: asadali.shahid@mail.polimi.it

2nd Loris Roveda
Istituto Dalle Molle di studi sull'Intelligenza Artificiale (IDSIA)
Scuola Universitaria Professionale della Svizzera Italiana (SUPSI)
Università della Svizzera Italiana (USI) IDSIA-SUPSI
Lugano, Switzerland
email address: loris.roveda@idsia.ch. ORCID: 0000-0002-4427-536X

3rd Dario Piga
Istituto Dalle Molle di studi sull'Intelligenza Artificiale (IDSIA)
Scuola Universitaria Professionale della Svizzera Italiana (SUPSI)
Università della Svizzera Italiana (USI) IDSIA-SUPSI
Lugano, Switzerland
email address: dario.piga@supsi.ch

4th Francesco Braghin
Department of Mechanical Engineering
Politecnico di Milano
Milano, Italy
email address: francesco.braghin@polimi.it

***Abstract*—Robots are nowadays increasingly required to deal with (partially) unknown tasks and situations. The robot has, therefore, to adapt its behavior to the specific working conditions. Classical control methods in robotics require manually programming all actions of a robot. While very effective in fixed conditions, such model-based approaches cannot handle variations, demanding tedious tuning of parameters for every new task. Reinforcement learning (RL) holds the promise of autonomously learning new control policies through trial-and-error. However, RL approaches are prone to learning with high samples, particularly for continuous control problems. In this paper, a learning-based method is presented that leverages simulation data to learn an object manipulation task through RL. The control policy is parameterized by a neural network and learned using modern Proximal Policy Optimization (PPO) algorithm. A dense reward function has been designed for the task to enable efficient learning of an agent. The proposed approach is trained entirely in simulation (exploiting the MuJoCo environment) from scratch without any demonstrations of the task. A grasping task involving a Franka Emika Panda manipulator has been considered as the reference task to be learned. The task requires the robot to reach the part, grasp it, and lift it off the contact surface. The proposed approach has been demonstrated to be generalizable across multiple object geometries and initial robot/parts configurations, having the robot able to learn and re-execute the target task.**

***Index Terms*—Reinforcement learning, intelligent robotics, object manipulation, Proximal Policy Optimization.**

45 pt
0.625 in
15.9 mm

I. Introduction

A. Context

Industry 4.0 [1] requires robots to achieve smart behaviors in dynamic and flexible environments. In fact, production plants are going to be transformed into digital work-places, interconnecting machine systems, humans, products, etc., to re-configure the production on the basis of the current needs. This new paradigm creates the necessity to have the robot manipulator able to self-adapt its behavior to face the assigned ever-changing tasks. It is, therefore, not anymore possible to pre-program all the possible scenarios. Thus, intelligence has to be embedded into the robotic system, to sense and analyze the working-scene and to take decisions for the correct task execution, facing safety, performance and production issues. Artificial intelligence and machine learning approaches find, therefore, a huge space in the robotics domain to achieve such ambitious goals. This contribution deals with the here described context, proposing an approach for learning a grasping task in a simulation environment, that can be transferred to the real robot for the real task execution. In such a way, the robot is able to safely learn and execute a specific application task, with an ability to generalize and rapidly adapt its behavior to new tasks. In the following, the state of the art related to the proposed context, together with the aim of the paper are detailed.

45 pt
0.625 in
15.9 mm

B. Related works

Analyzing state of the art related to task learning, the two most relevant approaches are: i) Learning from Demonstration (LfD) or Imitation Learning, and ii) Reinforcement Learning (RL). The main distinction between these approaches lies around the fact that whether or not human demonstrations in any form are exploited for learning behaviors. LfD aims to learn tasks based on demonstrated trajectories, whereas RL discovers the optimal behavior for a task through trial and error, employing a reward function that encourages the desired behavior. Another approach involves combination of both approaches to learn a task from human provided demonstrations and improve the behavior with RL. In [2], an algorithm was able to master the game of Go by first learning the competitive Go policy

The work has been developed within the project ASSASSINN, funded from H2020 CleanSky 2 under grant agreement n. 886977.

from an expert's demonstrations, and then improving that policy through RL. Considering i), LfD methods have been used to learn control commands for a robot from raw sensory signals [3], and extract a reward function from demonstrations [4], also known as inverse reinforcement learning. Considering ii), RL methods have been successfully applied in a wide variety of contexts, ranging from playing games [5] to robotic locomotion [6]. In robotics, RL has enabled robots to learn tasks, such as playing table tennis [7], flipping a pancake [8], aerobatic helicopter maneuvers [9] and general manipulation skills [10]. Earlier RL methods used low dimensional representation techniques such as movement primitives to solve control problems [8], [11]. Recently, RL has been used by exploiting the function approximation power of deep neural networks [12]–[14]. Such techniques, named as deep reinforcement learning, leverage the capabilities of neural networks in learning representations from high dimensional input data thus, making it possible to learn control in end-to-end manner.

Recent works have seen emerging use of deep reinforcement learning techniques for robot manipulation. Notably, guided policy search, a model-based approach, is used to train *visuomotor* (coordination between vision and control) policies directly from raw images [15]. While this method achieves impressive results on real world manipulation tasks, significant human involvement is required for data collection process. Model-based RL approaches generally learn dynamic models from data that generate trajectories to subsequently aid policy learning [16]. They aim to learn either a smooth global model [17] or local time varying linear dynamics models [18], [19]. In both cases, these methods struggle to learn policies in tasks that have inherently discontinuous dynamics and rewards as demonstrated by [20]. Another idea is to use large-scale data collection to learn control in a self-supervised manner [21], or through model-free RL [22]. However, such kind of data collection is not economically feasible, requiring multiple robots and months to collect.

To deal with the sample complexity, another line of work exploits simulation data to learn RL based control policies [13], [23]. Simulation enables fast and efficient comparison of design choices, since modern simulation engines allow orders-of-magnitude speed advantage against real time. Moreover, having access to the full state of the system in simulation, *e.g.*, part position, allow faster training of policies. Recent works have shown success in using simulation data to learn visuomotor RL policies [24], [25]. However, vision based learning demands an additional *sim-to-real* techniques with excessive computation requirements due to unrealistic rendering capabilities of simulation engines [26]. As an alternative, state space without any vision component serve as a sufficient representation of reality when simulation environment roughly matches the characteristics of real robot kinematics and manipulated part, and facilitates in learning RL policies with less computation and time requirements. Therefore, simulation based training of RL policies is well suited in the settings with constraints on real-data requirement and where simulated environment can adequately represent real conditions.

Considering grasping learning, most recent approaches involve modular pipelines and open-loop control of a planned grasp, *e.g.*, first estimating a grasp pose to predict the best grasping contact points and then using motion planning to reach the location [27], [28]. In contrast, the approach in this paper relies on reinforcement learning and neural networks, like [22], to enable dynamic closed-loop control for grasping. However, [22] use off-policy RL and multiple robots while also constraining the robot's work place, the approach in this paper uses on-policy RL without vision and without any constraints, thus, providing a more generic framework for task learning.

C. Paper contribution

Training in simulation is a practical approach for learning manipulation tasks with deep reinforcement learning [6], [23]. In this paper, the possibility of using simulation data to learn manipulation tasks in end-to-end manner with a model-free RL algorithm is investigated. Specifically, Proximal Policy Optimization (PPO) is elected to train the robot controller, due to its recent success on difficult control problems, *e.g.*, manipulation of a Rubik's cube with robot hand [29]. Therefore, the control actions (*i.e.*, the desired joint velocities and gripper's joint position) are learned in continuous spaces on the basis of a defined reward function, allowing to take into account the success of the task and its performance. The learned behavior can then be transferred to the real robot, to execute the target task. A grasping task involving a Franka Emika Panda manipulator has been considered as the reference task to be learned. The task requires the robot to reach the part (nominal part: cube of 6 cm length), grasp it, and lift it off the contact surface. The proposed approach has been demonstrated to be generalizable across multiple object geometries and initial robot/parts configurations. In fact, despite the task has been trained only on a single cube with no prior task information, the learning is able to generalize across different geometric shapes and sizes, minor changes in the position of the object to be manipulated, and across multiple initial configurations of the robot.

45 pt
0.625 in
15.9 mm

II. Problem formulation

The primary objective of this contribution is to autonomously learn the robot control actions for new tasks execution, without requiring any real data and/or human demonstrations, and no prior knowledge about system dynamics. The learned skill can then be transferred to the real robot, to execute the target task. To address this objective, large data is generated in simulation, learning to map observed states directly to commanded joint velocities. As real-world robotics problems are often best represented as continuous state and action spaces, the learning of the control actions is performed in continuous spaces. To do that, the Proximal Policy Optimization (PPO) algorithm is proposed to train the robot controller, defining a reward function to guide the learning taking into account the task success/failure and performance. The block-diagram describing the proposed approach is shown in Figure 1, and highlights

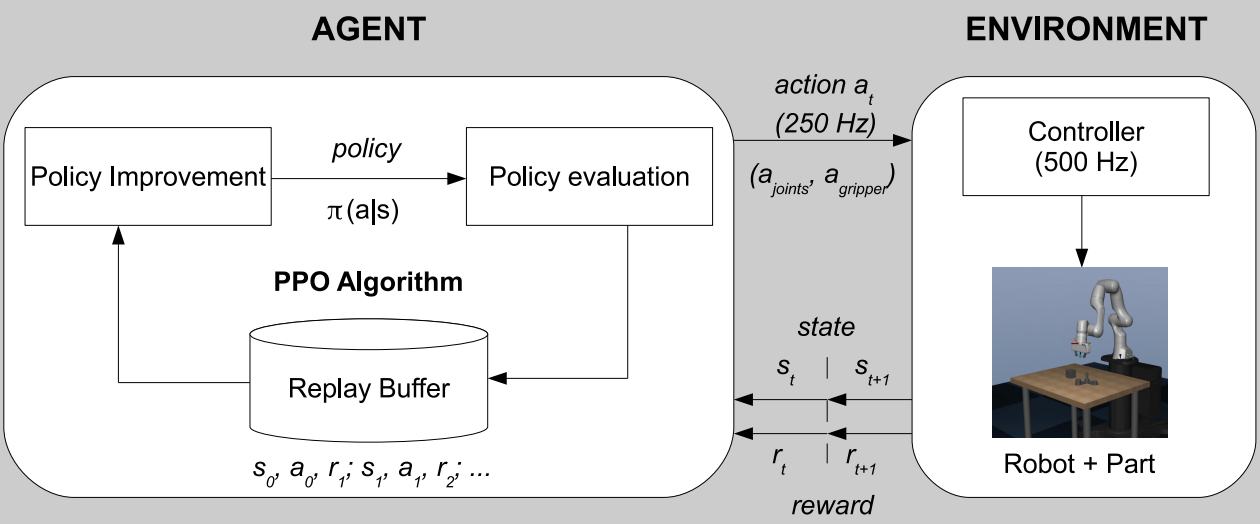

Fig. 1. Proposed learning schema exploiting PPO algorithm.

the implemented PPO algorithm and its connection with the simulation environment for learning purposes.

III. Methodology

A. Preliminaries

In this paper, a RL framework is considered, where an agent interacts with environment in discrete time steps. The RL problem is modeled as discrete time continuous Markov decision process (MDP) [30], consisting of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is continuous state space, $\mathcal{A}$ is continuous action space, $\mathcal{P}$ is environment transition dynamics defining the state transition probabilities for a given action $p(s_{t+1}|s_t, a_t)$, $\mathcal{R}$ is a reward function $\mathcal{R}(s_t, a_t) = r_t \in \mathbf{R}$, $\gamma \in [0, 1]$ is a discount factor determining the importance of future rewards relative to immediate reward. A policy π is stochastic that maps from states to distribution over actions $\mathcal{S} \rightarrow \mathcal{A}$. The objective of an agent is to learn a policy $\pi(a_t|s_t)$ that maximizes expected returns $\mathbf{E}[\sum_{t=0}^{H} \gamma r_t]$, where H is the horizon. In deep reinforcement learning, the policy is represented by a non-linear function approximator, a neural network parameterized by θ. The main goal then reduces to optimizing θ, such that the optimal behavior is guaranteed. RL methods fall in to 3 main categories: value-based, policy search and actor-critic [31]. Value-based methods learn a value function that implicitly derives a policy. Policy search methods, on the other hand, use parameterized policy and directly search for optimal policy parameters. The key idea of actor-critic approaches is to learn both the value function and the policy. The role of critic is, on the basis of learned value function, to critique the actions taken by an actor, which then updates it's policy parameters based on critic's feedback [32]. The proposed approach follows actor-critic style and learns both the parameters of policy and value function approximator.

B. Algorithm

In this paper, a model-free RL approach has been used as it eliminates the need for accurate dynamics model, which is often difficult to learn for discontinuous problems. The policy is trained with Proximal Policy Optimization (PPO) [33]. PPO is one of the most successful on-policy RL algorithms. It requires training two neural networks to simultaneously optimize stochastic policy and approximation of value function. The policy network maps states to Gaussian distribution over actions while the value network estimates the discounted sum of future rewards expected in a given state. The PPO algorithm is an approximate version of trust-region policy optimization (TRPO) [34] with first order gradients. The clipped objective in PPO improves the training stability by limiting the policy change at each iteration. The optimization objective in PPO is performed on the following surrogate loss function L_{PPO}:

$$\begin{aligned} L_{PPO} &= \mathbb{E}\left[\min\left(arg_1, arg_2\right)\right], \\ arg_1 &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t^{GAE}, \\ arg_2 &= clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) \hat{A}_t^{GAE}, \end{aligned} \tag{1}$$

where $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ denotes the probability ratio between the given action under current policy π_θ and the same action under old behavioral policy $\pi_{\theta_{old}}$ that was used to generate trajectory. $\hat{A}_t^{GAE}$is the advantage estimate computed using generalized advantage estimation (GAE) [13], and ϵ is a hyperparameter set to 0.2. The pseudo-code of selected PPO algorithm is shown in 1.

C. Network architectures

Both the policy and value networks are encoded by multi-layer perceptron (MLP). The policy network is a 3-layer MLP with size of 64 for both hidden layers and Rectified Linear Unit (ReLU) non-linearity. The output layer of a policy network has

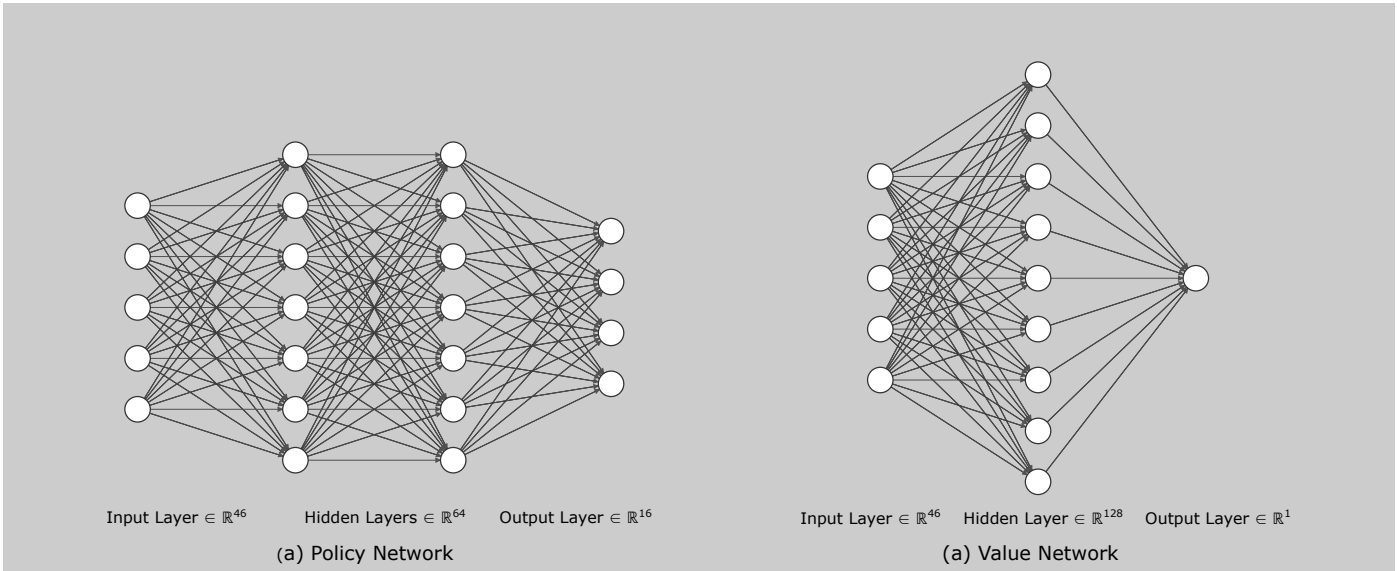

Fig. 2. Neural networks architecture.

16 units, producing the mean and the standard deviation for each action dimension. Before executing an action in the environment, *tanh* activation function is applied to enforce action bounds in range of $[-1, 1]$. The value network is represented as a 2-layer MLP that outputs a scalar value specifying the corresponding value of a state. The value network uses a hidden layer size of 128 units with ReLU non-linearity. All inputs fed to the policy and value network are normalized with running estimates of mean and variance. A high-level representation of network architectures is shown in Figure 2.

45 pt
0.625 in
15.9 mm

IV. Implementation and Evaluation of the Proposed Approach on Grasping Task Learning

A. Task description

The aim of the paper is to implement the described approach to autonomously learn a grasping task. The task consists of a robot interacting with a cube of nominal size 6 cm placed on a table. The goal of the robot is to successfully position its grip site around the cube, grasp it and then finally lift it off the contact surface. In order to evaluate the proposed approach on a grasping task learning, Franka Emika Panda, a 7-DoF torque-controlled robot is used as a robotic platform.

Algorithm 1 Proximal Policy Optimization (PPO) (adapted from [33]).

Randomly initialize policy parameters θ
Initialize replay buffer $\mathcal{B}$
for $iteration = 1, ..., N$ **do**
  for $episode = 1, ..., M$ **do**
    Generate a rollout following policy π_θ until H timesteps
    Store transitions(s_t, a_t, r_t) in $\mathcal{B}$
    Compute advantages $\hat{A}_t^{GAE}$ with GAE
  end for
  for $epochs = 1, ..., K$ **do**
    Sample mini-batch of N_b transitions from $\mathcal{B}$
    Optimize surrogate loss L_{PPO} w.r.t. θ
    $\theta_{old} \leftarrow \theta$
  end for
  clear $\mathcal{B}$
end for

B. Learning environment

The proposed learning environment is developed based on [35], exploiting MuJoCo physics engine [36] to simulate the physical system. MuJoCo enables fast and accurate simulation with contacts. The simulation environment is shown in Figure 3.

45 pt
0.625 in
15.9 mm

1) States and Actions: Both the state space and action space used in the proposed evaluation are continuous. State of the system consists of 2 main input modalities: robot proprioception information and object information. The proprioceptive data contains joint positions, velocities and end-effector position and orientation, thus making a 36-dimensional vector. Object

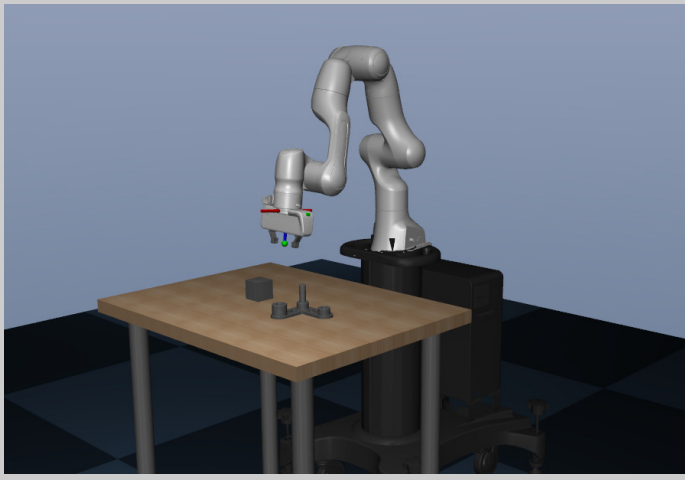

Fig. 3. Franka Emika Panda Robot in MuJoCo simulation environment.

information is 10-dimensional and includes cube's position, orientation and a relative position vector of cube and grip site. Actions are 8-dimensional representing the desired joint velocities and gripper joint's position. Hence, the learning agent learns the appropriate references for joints velocities and gripper's position to execute the grasping task. The robot is equipped with the joint torque actuators in simulation, therefore, a function to map desired joint velocities to joint torques is used:

$$\boldsymbol{\tau} = \mathbf{k}_v(\dot{\mathbf{q}}^d - \dot{\mathbf{q}}), \tag{2}$$

where $\dot{\mathbf{q}}^d$ and $\dot{\mathbf{q}}$ are desired and current joint velocities and

$$\mathbf{k}_v = diag(8, 7, 6, 4, 2, 0.5, 0.1)$$

is diagonal matrix of fixed proportional gains for joint trajectory-tracking purposes.

2) Control parameters: An important parameter that affects the trade-off between computational time and accuracy of the simulated task in MuJoCo is the simulation time step. The default value of 2ms is used, ensuring a good trade-off between simulation accuracy and stability. The controller of the Franka Robot operates at 500 Hz. Different values of the policy frequency (the rate at which policy output actions) have been tested (500 Hz, 250 Hz, 100 Hz), resulting in the best setting of 250 Hz. Since the robot controller generates commands at higher frequency than the policy, a linear interpolation is performed between each successive policy action.

C. Reward shaping

In order to guide policy learning and provide frequent feedback to the agent on appropriate behaviors, the proposed approach exploits the dense reward function. The task is split into two main phases of reaching and lifting, with distinct reward for each phase. The robot learns both of these phases simultaneously (*i.e.*, in a single iteration of a simulated task).

Considering the reaching phase, the reward given at time step t is composed of three terms:

$$r_t = w_{dist}(r_{dist}) + w_{vel}(r_{vel}) + w_{grip}(r_{grip}), \tag{3}$$

where r_{dist} is the distance reward computed using relative position of gripper site and object, r_{vel} is the velocity reward computed using end-effector velocity vector and r_{grip} is gripper open reward depending upon the action of gripper. In particular, r_{vel} and r_{grip} encourage the robot's end-effector to approach cube with small velocity and open gripper. All three contributions $r_{dist}, r_{vel}, r_{grip}$ are weighted with respective weights of 0.6, 0.3 and 0.1 and defined as:

$$\begin{aligned} r_{dist} &= 1 - tanh(\mathbf{p}_{grip} - \mathbf{p}_{cube}), \\ r_{vel} &= \begin{cases} 1 - tanh(abs(\mathbf{v}_{ee}/N_{dim}) & if\ \ r_{dist} < 8 \text{ cm} \\ 0 & otherwise, \end{cases} \\ r_{grip} &= \begin{cases} abs(action_{gripper}) & if\ \ action_{gripper} < 0 \\ 0 & otherwise, \end{cases} \end{aligned} \tag{4}$$

where $\mathbf{p}_{grip}$ and $\mathbf{p}_{cube}$ denote the position vector of gripper site and cube in world reference frame, $\mathbf{v}_{ee}$ is the end effector's velocity vector containing both linear and angular components with N_{dim} as 6, and $action_{gripper}$ specifies the action of gripper with negative values indicating that gripper is opening.

When the robot's grip site is in close proximity to the cube, the lifting phase is initiated. The distance threshold for switching is set to 2.5 cm. This is because nominal cube used for training measures as 6 cm and the robot should be able to grasp the cube when it's grip site is within cube's boundary. Considering the lifting phase, the reward is computed considering 5 contributions:

$$r_t = (r_{dist}) + r_{vel} + w_{grip}(r_{grip}) + r_{grasp} + r_{success}, \tag{5}$$

where r_{dist} and r_{vel} are same as in reaching phase, whereas r_{grip} is reversed now to give reward for closing the gripper. In addition, r_{grasp} is given if both fingers of the gripper are in contact with cube and $r_{success}$ rewards for successful completion of the task. Success is determined by examining if the gripper holds the cube above certain height. $r_{grip}, r_{grasp}, r_{success}$ in lifting case are defined as:

$$\begin{aligned} r_{grip} &= \begin{cases} abs(action_{gripper}) & if\ \ action_{gripper} > 0 \\ 0 & otherwise \end{cases} \\ r_{grasp} &= \begin{cases} 0.5 & if\ \ fingers\ \ in\ \ contact \\ 0 & otherwise \end{cases} \\ r_{success} &= \begin{cases} 2 & if\ \ z_{cube} > 0.1 + z_{cube_{initial}} \\ 0 & otherwise \end{cases} \end{aligned} \tag{6}$$

D. Results

PPO is applied to the task of lifting the cube above a certain height. The main aim is to analyze whether the proposed method can learn to accomplish this task and how well can it generalize to new situations that were not seen during training. The policy is trained for a total of 10 million time steps where each episode lasts for 600 steps, giving an agent approximately 2.5 seconds to accomplish the task. At the beginning of each episode, initial configuration of the robot and the cube is reset to fixed position. The simulation has been roughly sped-up 2000 x faster than real-time in order to train faster (requiring approximately 3 hours and a half of wall-clock time). The results of training are shown in Figure 4 and Figure 5, suggesting that the the agent has learned to perform the task successfully after 4 million steps.

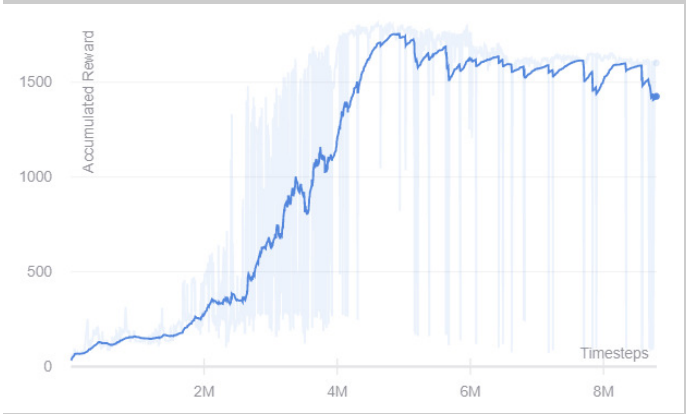

Fig. 4. PPO training results. The plot depicts the progression of mean accumulated reward for episodes.

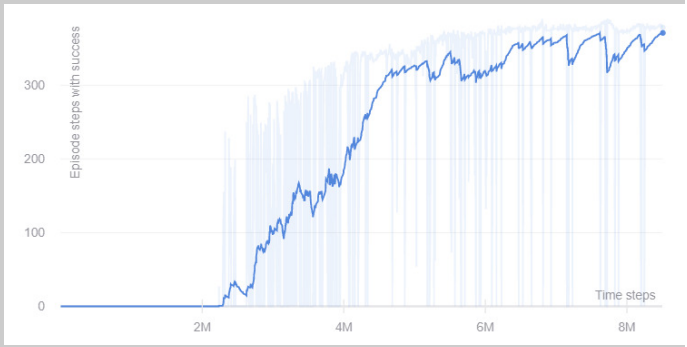

Fig. 5. PPO training results. The plot depicts the progression of mean success rate for episodes.

The generalisation capabilities and robustness of the learned neural network policy have been tested, considering nominal cube and different geometries to be grasped:

- nominal cube with size 6 cm;
- smaller cube with size 4 cm;
- cylinder with size 3 cm radius and 3 cm height;
- screw-driver.

10 test trials have been run and results are summarised w.r.t. successful completion of the task. The proposed model shows the success rate of 100% in all four cases. In last two cases, the policy's ability to grasp new shapes is tested by replacing the cube with a cylinder and a screw-driver placed nearly at the cube's original position.

Table II shows the results in which the original position of robot and/or of the cube has been changed. In the first case, a small random noise has been added to the initial joint positions of the robot at the beginning of each episode. The robot can still grasp and lift the cube in most cases, whereas in 2 failed trials, the robot could not successfully position the gripper around the cube and remained near the cube for the rest of episode. For the second case, the nominal position of the cube has been modified by ± 8 cm in order to test the policy's robustness to variation in part's position. For this specific case, 20 trails are performed, 10 for each direction and success rate is reported as the mean of 20 trials. The task has been completed successfully in most test runs, while sometimes the robot's actions results in failure. In these specific situations, the robot either grasped the cube at the edges resulting in unstable grasp or collided its gripper with the cube not being able to grasp successfully. In the last test, further evaluation of the potential for adapting the learned task model have been performed with significant changes in position of the cube. With 30 minutes of additional training, the resulting performance achieves 100% success, indicating that the policy can successfully reuse prior task model and quickly adapt the behavior in new situations.

V. Discussion and Current/Future Works

In this paper, an intelligent task learning has been formulated as a RL problem, demonstrating the possibility of learning low-level control actions purely from gathered experience in a simulated environment. Results show that it is possible to train continuous control actions based only on the state observations, and in a reasonable amount of time. Achieved results highlight that the learned policy performs well to new situations, and it can also adapt its learning to significant variations of the environment with slight amount of additional training. One of the limitations of current method is the sample efficiency. In the future, three main avenues are going to be faced. First, the learning of the same task using off-policy RL algorithm will be considered, in order to improve sample efficiency, as the agent will efficiently use past experience data instead of discarding it like in current method. Additionally, the transfer of the learned task on a real-robot will be implemented, re-sampling the learned control actions to satisfy the real-robot control requirements, to validate the proposed approach. Furthermore, safety mechanisms will be included into the defined reward function to encourage the agent to not take actions near the joint limits whilst avoiding contact with other objects in cluttered environment.

TABLE I
Evaluation trials for fixed positions.

Test	Object	Success rate
1.	Nominal cube	10/10
2.	Smaller cube	10/10
3.	Cylinder	10/10
4.	Screw-driver	10/10

TABLE II
Evaluation trials for varied positions.

Test	Variable	Amount	Additional training	Success rate
1.	Robot position	max 2%	No	8/10
2.	Cube position	± 8 cm	No	7/10
3.	Cube position	$+10$ cm	Yes	10/10

REFERENCES

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[3] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3758–3765.

[4] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 182–189.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[7] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[8] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *2010 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2010, pp. 3232–3237.

[9] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in neural information processing systems*, 2007, pp. 1–8.

[10] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4639–4644.

[11] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3828–3834.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[13] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[14] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[15] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[16] M. P. Deisenroth, G. Neumann, and J. Peters, *A survey on policy search for robotics*. now publishers, 2013.

[17] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[18] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[19] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, 01 2015.

[20] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3381–3388.

[21] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[22] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[23] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[24] R. Martín-Martín, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space. an action space for reinforcement learning in contact rich tasks," in *Proceedings of the International Conference of Intelligent Robots and Systems (IROS)*, 2019.

[25] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *arXiv preprint arXiv:1707.02267*, 2017.

[26] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[27] D. Morrison, A. W. Tow, M. Mctaggart, R. Smith, N. Kelly-Boxall, S. Wade-Mccue, J. Erskine, R. Grinover, A. Gurman, T. Hunn *et al.*, "Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7757–7764.

[28] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1–8.

[29] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[31] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[32] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement learning in continuous action spaces through sequential monte carlo methods," in *Advances in neural information processing systems*, 2008, pp. 833–840.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[34] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[35] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, "Surreal: Open-source reinforcement learning framework and robot manipulation benchmark," in *Conference on Robot Learning*, 2018, pp. 767–782.

[36] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.