In-hand manipulation via deep reinforcement learning for industrial robots

Leonardo V. O. Toledo, Gustavo J. G. Lahr, and Glauco A. P. Caurin

Abstract Robotics manipulation is still a challenge in many scenarios, specially when the orientation of the tool or part is determinant for task success. In-hand manipulation techniques substitutes re-grasping, saving time during application and unlocking many possible manipulation tasks. A study with pivoting was conducted, which consists in re-orienting the part around one rotational axis without dropping. Although the control of underactuated manipulators was applied in similar scenarios, changes in the task of the tool or part makes it hard to model. It gets more complex in industrial robots, which are position controlled and often the user does not have access to the dynamics parameters. Deep reinforcement learning has been successful with model free approaches as it learns the behavior of the whole system. An experiment for pivoting was conducted for part alignment to a desired angle. A robotic wrist with one degree of freedom and a parallel gripper was controlled in two different manners: position control (industrial robot) and torque control. Simulated experiments show that a position controlled robot is capable of executing the task, and as less relative movement is desired, smaller errors were achieved by the robot. Also, when tested with parts of different sizes, the robot was still capable to complete the task under the acceptable error, not reaching the permitted value only above a deviation of 20° from the initial angular position.

Key words: Robotics Manipulation, Pivoting, Deep reinforcement learning.

1 Introduction

Robotics manipulation has been increasing its applications due to more computational power, better mathematical approaches, and improved interaction theories [10]. The advance in new AI techniques allied to precise simulators may lead to robust and applicable alternatives in most diverse environments, such as field robots [17], industrial robotics [8] and dexterous manipulation [3]. Many tasks require picking an object and placing it in a different position. However, the initial grasp may produce some deviation from the desired gripping angle. This variation occurs from many factors, such as distinct initial positions from both gripper and object, changes in friction, a variation on the final position profile, and many others [9]. Solutions to this problem are re-grasping or the use of in-hand manipulation techniques.

Leonardo V. O. Toledo, Gustavo J. G. Lahr and Glauco A. P. Caurin
Sao Carlos School of Engineering, Av. Trabalhador são-carlense, 400, São Carlos, São Paulo, Brazil e-mail: {toledo.leo,gustavo.lahr}@usp.brandgcaurin@sc.usp.br

Re-grasping restarts the task and try a new pick to minimize the gripping error [11], which can be very time consuming or not possible. Meanwhile, in-hand manipulation aims to solve the problem when the object still on the gripper. Methods like the one described by Furukawa et al. [7] propose a human-like approach for in-hand re-grasping. Chavan-Dafle et al. [6] propose customized grippers to avoid releasing the tool. Although the positive efficiency, these methods require high mechanical complexity, which is expensive and time consuming for industrial tasks.

In this paper, we study the pivoting strategy as an in-hand technique for tool-part reorientation [2]. Pivoting an object consists of rotating it along a single axis to reorient it to the desired angle.It is very useful specially with parallel grippers, which have a broad use in industry. Control alternatives for pivoting would be the implementation of underactuated control of manipulators [14], as the robot would be seem as the actuated and the tool or part, not actuated. However, precise models of the robot or the parts are not easily obtainable in industrial assembly [16]. Industrial robots commonly have their dynamics not known by the user, so friction and size may vary in different scenarios. Also, it is also difficult to obtain a good friction model for part-tool and robot interaction, especially for robust simulations.

Deep reinforcement learning has presented itself as a good candidate to solve this problem. Very challenging applications were done by deep models in reinforcement setup, such as the in-hand cube manipulation with robotic hand [3] and industrial assembly tasks [8]. However, these setups are expensive and sensor rich, which is not always the case for industrial applications. Other in-hand manipulations have less complex setups, but they usually use torque controlled joints, and this option is not available in industrial robots [4].

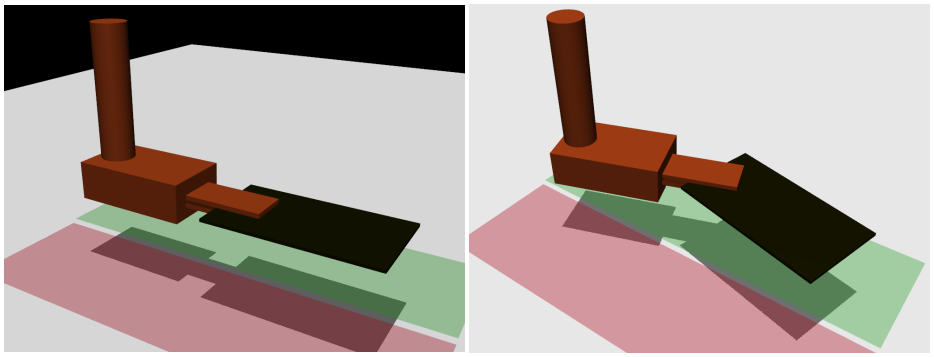
We present a deep reinforcement learning setup to learn in-hand manipulation for positioning part with a parallel gripper in industrial robots. Using Proximal Policy Optimization as reinforcement learning algorithm, with function approximation done by multi-layer perceptron, we compare the torque controlled approaches with the proposed position controlled and show the similar learning behavior. This indicates that it is possible to use deep reinforcement learning in industrial robots.

2 Modeling and Control

This section provides a formalization of the problem and describes the modeling and control. The goal is to perform the pivoting of a part from an initial angle to the desired angle as shown in Figure 1. A parallel gripper holds the part, while the arm can either rotate along a single axis generating inertial forces to move the part or control the holding pressure on it by moving its fingers, varying the normal force. We assume that the task to be solved starts already with the part grasped.

Deep Reinforcement Learning algorithms are good candidates to control the task. Their negative point is the time it takes to learn in real models, since steps like the reset of the robot on each episode may be very time-consuming. To deal with this setback, we propose a previous phase, using a simulation to speed up training. Further, the trained network can be transferred to real applications using transfer learning techniques [18].

In the following sections, we present a robust simulation model, build with Mujoco [19] and its training using the Proximal Policy Optimization [13] algorithm.



(a) (b)

Fig. 1: Rigid bodies model built in Mujoco. Image a) shows the system before a task has started and b) after it is completed

2.1 Rigid bodies modeling

We used Mujoco to build the rigid body model. The main body is composed of a box B_{main} that contains a revolution joint R, limited to $\pm 40°$ and uses a proportional gain of $Kp = 30$. Connected to B_{main}, there is a two finger gripper, modeled by a B_{upper} represents the upper and B_{lower} the lower one. B_{upper} contains a prismatic joint P, and thus controls the pressure over the part B_{part}, with a limited range of $\pm 2.5\ mm$ and $K_p = 80$. A free joint is attached to the part, meaning it is free to assume every possible position in space. Figure 3 gives the labels of each element of the model. Training was conducted with a part of dimensions $(L_{part}, b_{part}, h_{part}) = (215, 120, 4)$ mm and mass 0.82 kg.

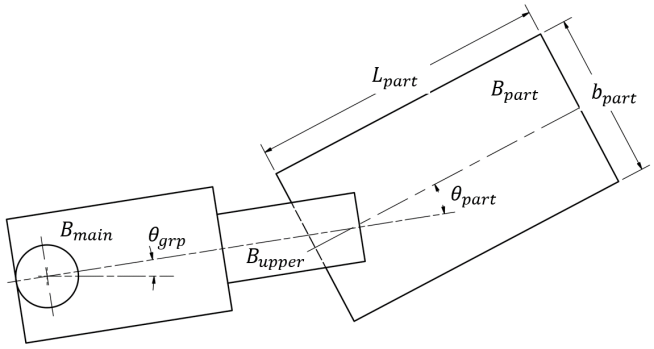


Fig. 2

Fig. 3: Schematic of the rigid bodies model

The system stands on a horizontal plane. The gravity acceleration used is $-9.81 m/s^2$. The sliding friction coefficient, acting along both axes on the tangent plane, is 0.75 for B_{part}, and 0.85 between B_{upper} and B_{lower}. The torsional friction, acting around the contact normal, has a value of 0.004 between each body.

To confirm the task proficiency for industrial robots, we used a position controlled actuator. For comparing purposes, we trained a similar model with the same parameters but with a torque controlled actuator using gear = 20.

2.2 Control Strategy

In recent years, reinforcement learning algorithms have been increasing its applications in robotics, demonstrating great potential for solving challenging decision-making problems. In this section, we provide proper background for Markov Decision Process (MDP) [15] and Proximal Policy Optimization (PPO) [13]. The goal of any reinforcement learning algorithm is to solve a MDP [15], defined as a 4-dimensional tuple (S, A, P_a, R_a):

- S represents a finite space of states;
- s_t is the observed space at time t;
- A represents a finite space of actions;
- a_t is the action taken at time t;
- $P_a(s_t, s_{t+1})$ is a transition function, denoting the probability of an action a_t at time; t lead from a state s_t to s_{t+1} at time $t+1$;
- $R_a(s_t, s_{t+1})$ gives the reward of transitioning to a state s_{t+1} from a state s_t.

The core problem of a MDP is to find a policy π witch defines the action $\pi(s_t)$ that our agent will choose at state s_t. To solve the MDP problem, we address the Proximal Policy Optimization (PPO), a *on-policy* [15] algorithm with the same principles of Trust Region Police Optimization (TRPO) [12]: taking the biggest possible improvement step without causing a performance collapse. Although both methods have similar efficiency, PPO makes things simpler by solving the problem with first-order equations, while TRPO makes use of second-order methods. PPO updates policies via (1).

$$L^{CLIP}(\theta) = E_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \ , \tag{1}$$

Where,

- θ is the policy parameter;
- E_t denotes the empirical expectation over timesteps;
- A_t is the estimated advantage at time t;
- ϵ is a hyperparameter (usually 0.2);
- r_t is the probability ratio under the new and old policies, given by (2).

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \ , \tag{2}$$

The term $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t$ in Equation (1) works as filter of the probability ratio, removing the incentive of moving r_t outside of $[1-\epsilon, 1+\epsilon]$, therefore preventing possible policy degradation.

3 Learning

The learning is built upon Gym [5] and Spinning Up [1], both modules created by OpenAI, with the purpose of facilitate the use of Reinforcement Learning algorithms.

The *Observation Space* is composed by the measured variables defined by (3).

$$s_t = [\theta_{part} - \theta_{tgt}, d_{dist}, \dot{\theta}_{grp}, \dot{\theta}_{part}, drop] \tag{3}$$

- θ_{part} - part's angle with relation to the gripper;
- θ_{tgt} - target angle;
- d_{dist} - distance between B_{upper} and B_{lower};
- $\dot{\theta}_{grp}$ - gripper's velocity;
- $\dot{\theta}_{part}$ - part's velocity with relation to the gripper;
- *drop* - Boolean variable indicating if the part dropped from the gripper.

The first parameter, $\theta_{part} - \theta_{tgt}$, gives us the current distance from the goal, allowing the network to generalize different policies for different distances. The second one, d_{dist}, controls the pressure over the part, exerted by B_{upper} and B_{lower}.The third and fourth, $\dot{\theta}_{grp}$ and $\dot{\theta}_{part}$, gives the correct velocity values that actually generate the expected dislocation of the part. The *drop* allows the agent to tell if it is actually holding the part or not.

The *Action Space* at each time step t is given by (4).

$$a_t = [\theta_{grp}, d_{dist}] \tag{4}$$

Where θ_{grp} represents the gripper's position. With these 2 actions our agent is able to accomplish the objective. Our method validates it's proficiency for industrial assembly, where more complex solutions may be impracticable regardless the success rate.

The reward works with 3 distinct rules, shown in (5). The base rule is present from the beginning of the simulation until the part reaches the success zone, that is, the region $\theta_{tgt} \pm \Delta_\theta$, where Δ_θ is the acceptable error. Once the part is at the success zone $\eta = 2$, otherwise $\eta = 1$. This generate extra incentive of remaining inside this region. To minimize the difference from θ_{tgt} and make sure of the stability, the agent must stay in the success zone for 120 time steps. The third and last rule applies after reaching the time restriction. On it, the episode ends and the agent receives $R_t = +10$.

$$r_t = \begin{cases} -\frac{|\theta_{part} - \theta_{tgt}|}{100\eta}, & \theta_{part} \in (-\infty, (\theta_{tgt} - \Delta_\theta)) \cup ((\theta_{tgt} + \Delta_\theta), +\infty) \\ 10, & \theta_{part} \in [(\theta_{tgt} - \Delta_\theta), (\theta_{tgt} + \Delta_\theta)] \end{cases} \tag{5}$$

The success zone is given by:

$$\Delta_\theta = \begin{cases} \left(\frac{|\theta_{tgt}|}{7} + 0.3\right), & -6 \leq \theta_{tgt} \leq 6 \\ \min\left(\frac{|\theta_{tgt}|}{7}, 3\right), & \theta_{tgt} < -6 \cup \theta_{tgt} > 6 \end{cases} \tag{6}$$

Using a variable error, instead of fixed 3°, for instance, forces the agent to be more precise for smaller angles and allows it to be able to adequate for larger ones. The limit of 3° holds this adequacy from going too far and undermine the simulation. In Section 4, Figure 5 illustrates the definition above.

4 Results

4.1 Training

The training occurred within 2000 episodes. Each episode has the maximum duration of 4000 time steps. The number of time steps per epoch was set to 8000. A multi-layer perceptron network was used for

function approximation. Various architectures were tested, but it responded it's best with 3 hidden layers of 32, 24 and 10 neurons, respectively. We used the discount factor of $\lambda = 0.99$.

Figure 4 shows the mean reward of the position controlled actuator in blue and torque controlled in red. The system converges to the object faster in the position controlled model and remains more stable during the whole training. It is worth mentioning that all the conditions were kept the same for both models, changing only the used actuator.

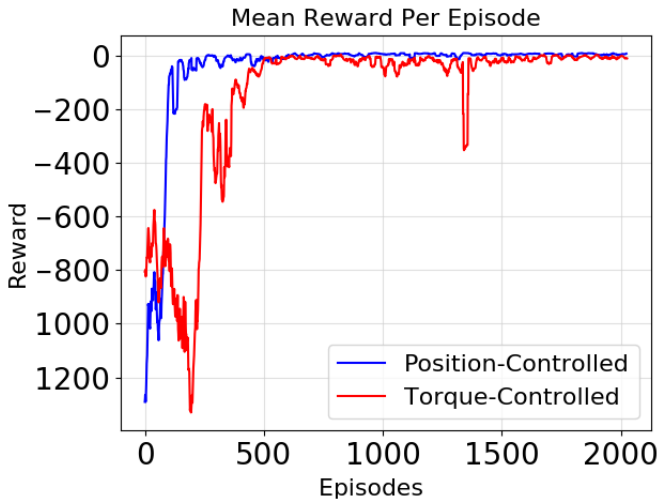


Fig. 4: Mean Reward per Episode during Training phase

4.2 Experiments

In this section, we present the results for 2 different comparisons. The first one compares practical results of both presented actuators, and the second one reveals the robustness of our model for different parts.

Since the reward plot in (4) shows a mean value, it can hide some fails. The model may work very well for some range of angles and fail completely in other range. To analyze that, Figure 5 shows the error in degrees when trying to reach a target angle. The green area represents the success zone, defined in Equation (6). For each target angle, 30 runs were done, and then we extract the mean value of it.

All the results so far were collected using a single part. To confirm the generalization of the model, we must verify the task execution on different parts. Therefore, two extra parts were tested: part 2 with dimensions 120**x**100**x**4 *mm* and 0.38 kg, and part 3, 180**x**220**x**4 *mm* and 1.27 kg. The friction was kept the same as the previous experiments.

Figure 6 presents the error for all parts over the target angle. Meanwhile the trained part was capable to achieve all desired angles under the acceptable errors, due to its high fidelity to original data, the other had problems and started to deviate with more than 20° from initial position, either being bigger (part 3) or smaller (part 2). This happens due to the fact that the manipulator needs to move faster to achieve the target angle, thus applies higher accelerations, leading to not negligible inertial components, which is harder to generalize. This behavior is function of the inertia of the parts, but also to the friction in interaction between grippers and part, that is more complex to model and has a wide range of values.

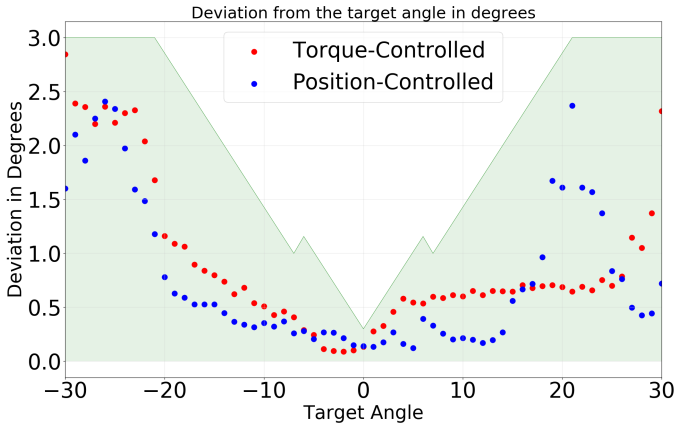


Fig. 5: Deviation from the target angle when in degrees with 2 different actuators

For most of angles range, different parts confirm the model flexibility, while it is possible to observe failures for larger angles. In future work, we consider adding different parts during the training process and include variables like mass and part dimensions to the observation space.

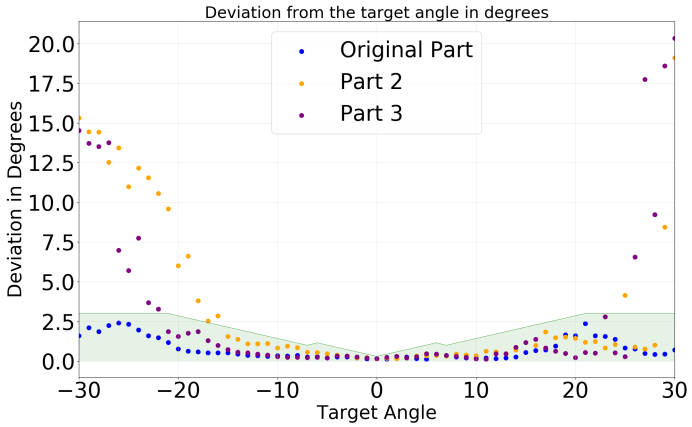


Fig. 6: Deviation from the target angle in degrees for 3 different parts

5 Conclusions

This works propose the use of reinforcement learning algorithms to solve in-hand manipulation problems for industrial robots, focused on pivoting. Pivoting consists on the reorientation of the object by generating inertial forces to move the part. The model was implemented using a precise physics simulator, Mujoco. The learning algorithm applied to solve the task was Proximal Policy Optimization and the actuation was executed by a position controller, present in ubiquitously present in industrial automation systems, including robot manipulators.

The model achieved a high level of success for position and torque actuators, corroborating to it's robustness. All tests with the training part achieved small errors. When evaluated for generalization with different parts, the trained model was able to generalize for angles closer to the initial one, i.e., range of [-20°, 20°]. Errors larger than the acceptable were verified outside the mentioned range.

On future work, the implementation of a variety of parts during training and the inclusion of it's parameters in the observation space suggest a possible improvement in the generalization. Also, we would like to evaluate the effect of friction during training and generalization of the parts. To test the robustness of the model, the learning agent will be transferred to a real simulation on a industrial robot, with the help of transfer learning techniques.

Acknowledgements The authors would like to acknowledge CNPq for the processes 314936/2018-1 and 141395/2017-6, São Paulo Research Foundation (FAPESP) grant 2017/01555-7, and University of São Paulo for USP-PRP Call 668. This study was partially funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001.

References

1. Achiam, J.: Spinning Up in Deep Reinforcement Learning (2018)
2. Aiyama, Y., Inaba, M., Inoue, H.: Pivoting: A new method of graspless manipulation of object by robot fingers. In: Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93), vol. 1, pp. 136–143. IEEE (1993). DOI 10.1109/IROS.1993.583091. URL `http://ieeexplore.ieee.org/document/583091/`
3. Andrychowicz, O.M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., Zaremba, W.: Learning dexterous in-hand manipulation. The International Journal of Robotics Research **39**(1), 3–20 (2020). DOI 10.1177/0278364919887447
4. Antonova, R., Cruciani, S., Smith, C., Kragic, D.: Reinforcement learning for pivoting task. arXiv preprint arXiv:1703.00472 (2017)
5. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. CoRR **abs/1606.01540** (2016)
6. Chavan-Dafle, N., Mason, M.T., Staab, H., Rossano, G., Rodriguez, A.: A two-phase gripper to reorient and grasp. IEEE International Conference on Automation Science and Engineering (2015). DOI 10.1109/coase.2015.7294269
7. Furukawa, N., Namiki, A., Taku, S., Ishikawa, M.: Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system. IEEE International Conference on Robotics and Automation (2006). DOI 10.1109/robot.2006.1641181
8. Luo, J., Solowjow, E., Wen, C., Ojea, J.A., Agogino, A.M., Tamar, A., Abbeel, P.: Reinforcement learning on variable impedance controller for high-precision robotic assembly. In: 2019 International Conference on Robotics and Automation. Montreal (2019)
9. Mason, M.T.: Mechanics of Robotic Manipulation, 1st edn. MIT Press, Cambridge (2001)
10. Mason, M.T.: Toward Robotic Manipulation. Annual Review of Control, Robotics, and Autonomous Systems **1**(1), 1–28 (2018). DOI 10.1146/annurev-control-060117-104848
11. P. Tournassoud, T.L.P., Mazer, E.: Regrasping. IEEE International Conference on Robotics and Automation (1987)
12. Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P.: Trust region policy optimization (2015)
13. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
14. Siqueira, A., Terra, M.: Nonlinear and Markovian $\mathcal{H}_\infty$ Controls of Underactuated Manipulators. IEEE Transactions on Control Systems Technology **12**(6), 811–826 (2004). DOI 10.1109/TCST.2004.833626
15. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. A Bradford Book (2018)
16. Swevers, J., Verdonck, W., de Schutter, J.: Dynamic Model Identification for Industrial Robots. IEEE Control Systems **27**(5), 58–71 (2007). DOI 10.1109/MCS.2007.904659
17. Tao, Y., Zhou, J.: Automatic apple recognition based on the fusion of color and 3D feature for robotic fruit picking. Computers and Electronics in Agriculture **142**, 388–396 (2017). DOI 10.1016/j.compag.2017.09.019
18. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research **10**(1), 1633–1685 (2009)
19. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033. IEEE (2012). DOI 10.1109/IROS.2012.6386109