HAL
open science

Reward-Based Environment States for Robot Manipulation Policy Learning

Cédérick Mouliets, Isabelle Ferrané, Heriberto Cuayáhuitl

To cite this version:

Cédérick Mouliets, Isabelle Ferrané, Heriberto Cuayáhuitl. Reward-Based Environment States for Robot Manipulation Policy Learning. NeurIPS Workshop on Deployable Decision Making in Embodied Systems (DDM 2021), University of Toronto, Dec 2021, Toronto, Canada. pp.1-6. hal-03631670

HAL Id: hal-03631670

https://ut3-toulouseinp.hal.science/hal-03631670

Submitted on 5 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reward-Based Environment States for Robot Manipulation Policy Learning

Cédérick Mouliets
Engineering school UPSSITECH
University of Toulouse III, France
cederick.mouliets@univ-tlse3.fr

Isabelle Ferrané
IRIT, Paul Sabatier University
CNRS, Toulouse, France
isabelle.ferrane@irit.fr

Heriberto Cuayáhuitl
School of Computer Science
University of Lincoln, UK
hcuayahuitl@lincoln.ac.uk

Abstract

Training robot manipulation policies is a challenging and open problem in robotics and artificial intelligence. In this paper we propose a novel and compact state representation based on the rewards predicted from an image-based task success classifier. Our experiments—using the Pepper robot in simulation with two deep reinforcement learning algorithms on a grab-and-lift task—reveal that our proposed state representation can achieve up to 97% task success using our best policies.

1 Introduction

Teaching a robot new skills easily, quickly and reliably is one of the main goals of robotics and AI. In that way, robots could be of great assistance at home (for example carrying out cleaning or cooking tasks), in factories (carrying out assembly or quality check tasks during production of products), or in public spaces (for instance serving food/drinks or playing games), among other scenarios [1, 3, 7]. Some of the requirements for such an ambitious scientific and engineering goal include understanding robot commands from a human instructor, understanding of the environment and task, acquisition and processing of visual (and audio) perceptions, characterisation of the learning task, ability to select actions for achieving the task, and ability to measure performance, among others. Those requirements and the fact that the world situations are always different, make the problem of skill acquisition of a tall order. The fields of Deep Learning and Reinforcement Learning have been suggested as a promising framework to enable the requirements above. This implies the creation of an agent with a concrete set of states, actions, rewards, and a simulator or logged data. Training a robot means finding the policy that maximises the long-term reward signal via trial-and-error search [10, 14].

The learning scenario of focus in this paper is illustrated in Figure 1, where the Humanoid robot Pepper has to carry out a manipulation task in a simulated environment. We focused particularly on a classic task in robotics: grab-and lift. The robot has to grab an object and lift it without dropping it by using reinforcement learning. The novel aspect in this architecture is the use of compact environment states. In contrast to previous works that use states with learnt features from visual images [13, 11] (not as compact as ours), we use the last N=15 rewards to represent our proposed states. To our knowledge, this approach has not been studied before. Intuitively, those reward-based states can be seen as describing a sequence of task success scores—where the scores grow as the task progresses from a time step to the next.

NeurIPS Workshop on Deployable Decision Making in Embodied Systems, 2021

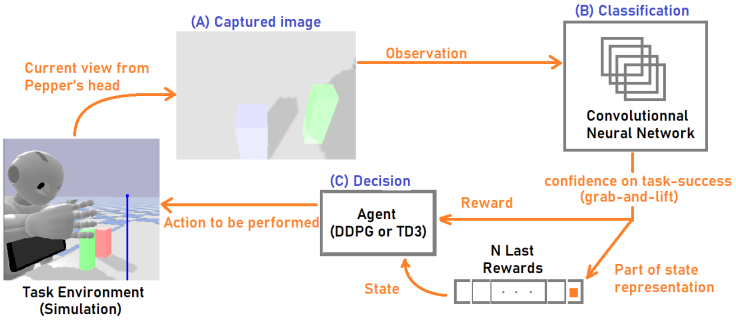


Figure 1: Cycle of the proposed learning agent

2 Related Work

Visual reward learning has been considered as a promising solution for equipping robots with learnable manipulation abilities [13, 11, 6]. This approach measures the success of experiences via reward in robotic tasks by estimating the probability of success from realistic input images [8]. While [13] focus on standard Convolutional Neural Networks (CNNs), [11] extend CNN-based architectures with additional paths on the neural network to add more information for better predictions and therefore calculating visual rewards more reliably. These works tested DRL (Deep Reinforcement Learning) agents in classic tasks using a variety of reinforcement algorithms such as DDPG, TD3 or SAC. To do so, different dense and sparse rewards have been investigated, where dense rewards have been found as a more promising alternative than sparse rewards.

Previous works in DRL for robot manipulation such as [4, 9, 5, 13] represent environment states mostly based on image-based features, where raw images are digested by CNNs in order to learn compact representations that can be used for policy learning. But the idea of using rewards as environment states or part the environment states has not really been studied. This highlights the need for a comparison of different state representations, which we identify as a research gap in this paper. In our case, we focus our study on comparing image-based against reward-based state representations.

3 Research Methods and Results

3.1 Methodology

We study the overall problem of how to train robot manipulation policies using reinforcement learning in a simulated environment. Given a state-of-the-art algorithm and a defined set of continuous actions, we adopted a reward function based on the success classifier approach as in [13, 11]. To study our proposed state representation, we relied on the following methodology.

1. Create an environment for the grab-and-lift task.
2. Create a dataset of success and non-success images.
3. Train a probabilistic task success classifier (to be used as a reward function).
4. Define reward-based states and action representations.
5. Determine the policy learning algorithm with support for continuous actions.
6. Train policies using the states, actions and rewards specified above.
7. Test the learnt policies using evaluation metrics such as avg. reward and avg. task success.

We used the qiBullet robot simulation environment [2], which is powered by the Bullet physics engine and the PyBullet real-time physics simulation framework. To create our simulation environment, we used Unified Robot Description Format (URDF) files for creating cube and table objects as illustrated in Figure 1—the task environment component. To make the robot interact with the cube objects, we specified friction, dampening and stiffness coefficients in the URDF files and put low weights (100g) so the cubes can be lifted. The images for visual perceptions are generated by the 2D camera on the mouth of the robot with resolution K_{QVGA}=(320,240).

3.2 Visual Reward Classifier and Policy Learning

Instead of manually engineering a reward function, we use learnt rewards from visual observations. To do so, we employ an image-based binary classifier to predict task success (CNN-based). The goal is to generalise situations on the simulation environment in a data efficient manner. From the front camera, the robot only sees two cubes on a table. 'Success' images show the robot's hand with the cube grabbed, from different angles, in front of the camera, while 'Non-Success' images show the cube outside the robot's hand. We randomised the arm positions (on or away from the cube) and generated a total of 6400 labelled images. This dataset of images was created from 10 balanced capture sessions, each session consisting of 200 success images and 440 non-success ones. From one capture session to another, we modified the robot's right arm to introduce variability in its work space. Non-success images are more present because they are more likely to be encountered. The neural architecture of our reward predictor is based on Tensorflow's image classifier [1].

To learn manipulation policies, we experimented two actor-critic algorithms for decision-making with continuous actions: DDPG [9], and TD3 [5]. We used the Stable-Baselines3 framework[2] with default hyperparameters to implement our policies and experimented different state representations. Regarding actions and since the proposed policies only use the right arm of the robot, we focused on four joints: *RShoulderPitch, RElbowRoll, RHand, and RShoulderRoll*. Those actions are generated by rotating at a constant speed the angle of each corresponding joint between 0° and 180°. A typical learning episode from Figure 1 is as follows. (A) The robot captures an image of the current situation from the simulated environment. (B) This image is sent to the pre-trained classifier which outputs a confidence score from its last Softmax layer. This score will be considered both as the reward as well as a part of the state representation. (C) Finally and via a deep reinforcement learning algorithm, the agent will choose an action that combines the four joints previously mentioned.

3.3 Reward Classification and Manipulation Results

We split our dataset of success/non-success images (6400 in 10 demos) into three subsets: 8 demos for training, 1 demo for validation and 1 demo for testing. We experimented different configurations in terms of number of epochs and batch size. Our results report that all of our 6 classifiers achieve high performance in terms of classification accuracy (98% average accuracy on test data-set) and we selected our best model for predicting visual rewards. Each classifier provides a confidence score of the success of the task which implies that the closer the arm is to the cube, the greater the confidence. Each classifier has been trained and tested using the same dataset split. The high performance is presumably due to the much simpler simulation environment than the real world. The classifier prediction time per image varied between 5.2 and 5.6 milliseconds[3].

We trained DDPG policies with six different state representations and compared against their corresponding TD3 policies. These state representations include three different image resolutions (320x240, 160x120, 80x60), Imagenet-based pre-trained embeddings[4] and PCA-based (Principal Component Analysis) images with 50 components [12]. Our goal was to improve them using our proposed compact representation based on the last 15 visual rewards (predicted from 160x120 images).

[1] `https://www.tensorflow.org/tutorials/images/classification`

[2] `https://github.com/DLR-RM/stable-baselines3`

[3] PC specs: 32Gb RAM, Intel Core I7-8650U CPU

[4] `https://github.com/jaredwinick/img2vec-keras`

Table 1: Best Policy Learning Results

Reinforcement Agent	DDPG | TD3	DDPG | TD3
State Representation	Avg. Reward	Avg. Task Success (%)
Pixels 320X240	2.000 | 2.100	76.60 | 78.11
Pixels 160x120	**2.500** | 1.800	80.09 | 78.34
Pixels 80x60	1.600 | **2.500**	28.40 | 88.04
Image embeddings	0.003 | 0.001	$2.7e^{-5}$ | $1.8e^{-4}$
PCA-based images	1.900 | 1.000	11.33 | 57.80
Last 15 rewards	1.834 | 2.326	**97.04** | **96.78**

PCA and image embeddings enable the significant reduction of amount of information contained in an image. While with PCA we reduced the amount of features to 50 components, with image embeddings we obtained vectors of 2048 dimensions.

Our learnt policies take on average 4:45 hrs to train and Table 1 shows their performance out of five runs. While at first sight it looks like TD3 outperforms DDPG in terms of average rewards, that is not the case in terms of average task success. The best DDPG policy obtained a task success of 97.04% and the best TD3 policy obtained 96.78% of task success as shown in Figure 2. Thus, both learning algorithms can achieve similar performance—but their best policy must be found from multiple runs. The policies were trained and tested for 10000 time steps. If the robot could not grab the cube after 50 actions, we considered the task as a failure and started over another episode as shown in Fig 2. Manual inspection showed that when an agent finishes episodes sooner than later, it obtains less rewards. Therefore, Average Task Success is a more reliable metric in our case. Our best policy uses DDPG with states including the last 15 rewards—a reward-based representation and the most successful in our comparative study. See episode examples in Figures 3 and 4 with successive steps.

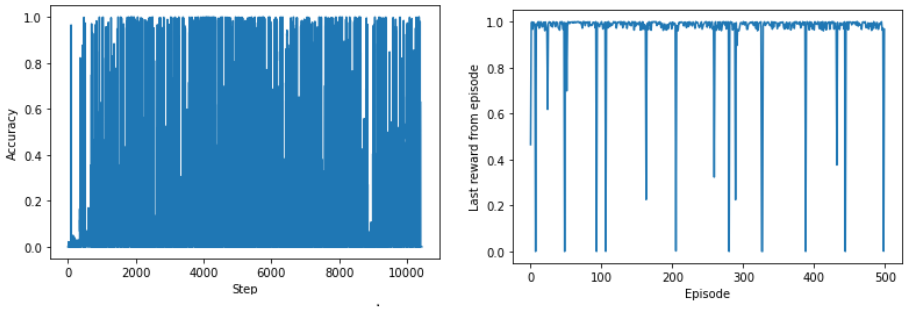
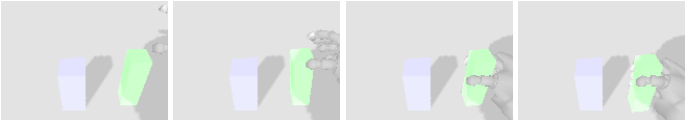
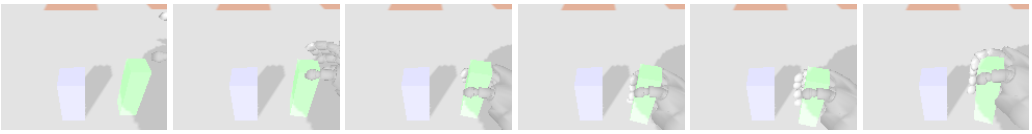


Figure 2: Training and testing of TD3 with 160*120 images and 50 actions a step



(a) r1=0.0001 (b) r2=0.0023 (c) r3=0.3486 (d) r4=0.9897

Figure 3: Example 4-step episode from initial to goal state, [captions]=probabilistic success rewards



(a) r1=0.0001 (b) r2=0.0144 (c) r3=0.8827 (d) r4=0.9463 (e) r5=0.6153 (f) r6=0.9928

Figure 4: Example 6-step episode from initial to goal state, [captions]=probabilistic success rewards

4 Discussion and Concluding Remarks

This paper proposes a novel methodology for learning manipulation policies based on states represented with sequences of observed rewards derived from a probabilistic task success classifier. Experimental results using simulated interactions in a grab-and-lift task show that our proposed state representation can be useful to learn policies with high-task success (up to 97%). Although other representations need to be investigated further such as tailored image embeddings or pre-trained embeddings with adaptation, our experiments show that it is indeed possible to achieve high performance with reward-based states instead of image-based states. Combining image-based and reward-based representations is an interesting strand of work that can be explored in the future. Two challenges that remain to be investigated with priority are how to efficiently create datasets for training task success classifiers, and how to safely deploy manipulation policies (learnt in simulation) in the real world.

References

[1] N. Akalin and A. Loutfi. Reinforcement learning approaches in social robotics. *Sensors*, 21(4):1292, 2021.

[2] M. Busy and M. Caniot. qiBullet, a bullet-based simulator for the pepper and NAO robots. *CoRR*, abs/1909.00779, 2019.

[3] H. Cuayáhuitl. A data-efficient deep learning approach for deployable multimodal social robots. *Neurocomputing*, 396, 2020.

[4] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *CoRR*, abs/1509.06113, 2015.

[5] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, volume 80, 2018.

[6] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, and G. D. Hager. "Good Robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer. *IEEE Robotics and Automation Letters*, 5(4), 2020.

[7] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *J. Mach. Learn. Res.*, 22, 2021.

[8] S. Kumra, S. Josh, , and F. Sahin. Learning robotic manipulation tasks through visual planning. *arXiv:2103.01434 [cs]*, 2021.

[9] 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. In *ICLR*, 2016.

[10] 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*, 10(1), 2021.

[11] A. Mohtasib, G. Neumann, and H. Cuayáhuitl. A study on dense and sparse (visual) rewards in robot policy learning. In *TAROS*, 2021.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12, 2011.

[13] A. Singh, L. Yang, C. Finn, and S. Levine. End-to-end robotic reinforcement learning without reward engineering. In *RSS*, 2019.

[14] T. Zhang and H. Mo. Reinforcement learning for robot research: A comprehensive review and open issues. *International Journal of Robotics Research*, 18(3), 2021.