Feature-Space Reinforcement Learning for Robotic Manipulation

Ralf Gulde, Khanh Nguyen, Marc Tuscher, Oliver Riedel and Alexander Verl

November 29, 2023

Task-Space Reinforcement Learning for Robotic Manipulation

Ralf Gulde, Khanh Nguyen, Marc Tuscher, Oliver Riedel and Alexander Verl

Abstract— **Reinforcement Learning (RL) has gained popularity for developing intelligent robots, but challenges such as sample inefficiency and lack of generalization persist. The choice of observation space significantly influences RL algorithms' sample efficiency in robotics. While end-to-end learning has been emphasized, it increases complexity and inefficiency as the agent must re-learn forward and inverse kinematics. To address these issues, we propose a straightforward approach that utilizes readily available control techniques, such as forward and inverse kinematics, to capitalize on domain knowledge. Our approach involves enhancing the observation space with task-space features and utilizing task-space inverse kinematics. Our contributions include a proposal for mathematical formulation and a framework for RL algorithms in robotics.**

Keywords: reinforcement learning, robotics application, task space representation

I. Introduction

In recent years, Reinforcement Learning (RL) has gained popularity as a technique for developing intelligent robotic systems. RL algorithms enable robots to learn from experience and optimize policies through trial and error, offering the promising potential for learning complex tasks and operating effectively in dynamic environments. As a result, Reinforcement learning has been implemented to solve various robotics applications, such as robot manipulation, grasping, locomotion, or autonomous driving.

Despite the notable advancements in RL with robotics, researchers still face several challenges and open questions. These challenges may include the requirement for efficient exploration in large state spaces, the need to learn from small amounts of data, the difficulties in dealing with non-stationary environments, and the necessity of learning from multiple agents [1]. These problems can be categorized into two significant issues: sample inefficiency and lack of generalization. Therefore, it is crucial to address these issues to enhance the robustness and effectiveness of RL algorithms for robotics applications. In this regard, the choice of observation space has been identified as a critical factor that significantly influences the sample efficiency of RL algorithms in robotics [2], making it a critical research area for advancing the state-of-the-art RL for robotics.

Recent advancements in reinforcement learning for robotics have emphasized end-to-end learning, where no prior knowledge of the system is used during training. As a result, the agent is required to re-learn forward and inverse kinematics through interaction with the environment, which has led to increased problem complexity and high sample inefficiency. In contrast, we propose a straightforward approach that utilizes readily available control techniques such as forward and inverse kinematics, which capitalizes on the domain knowledge of the problem.

Our approach involves enhancing the standard observation space with robotics task-space feature sets and utilizing task-space inverse kinematics to calculate the desired action for the robot. These features may include the robot's end-effector positions, velocities, or more complex features such as the distance between two shapes or accumulated collisions. The augmented observation space provides more valuable information to the RL agent, alleviating the need to learn everything from scratch. Additionally, employing task-space inverse kinematics eliminates the need for the robot to re-learn the IK process, thereby reducing sample inefficiency.

Our key contributions are:

- A mathematical formulation integrates the manipulation scene's arbitrary complex task-space features (accumulated collisions, oppose, etc.) into the vanilla Markov Decision Process.
- A framework applies the mentioned mathematical formulation into Reinforcement Learning algorithms for robotics applications.

II. Related Work

Deep Reinforcement Learning for Robotics Manipulation Tasks. Ni et al. proposed a technique using Deep Reinforcement Learning to train Robotic Arm Movement with the DDPG algorithm [3]. The authors experimented with this method on two tasks, Robot Reacher and Pick-And-Place, showing a success rate above 80% with only 6000 to 8000 episodes. Saeed et al. utilized the DDPG algorithm with Hindsight Experience Replay (HER) to address the problem of sparse rewards for robotic hand manipulation [4]. The authors benchmarked three policy gradient algorithms (DDPG, DDPG+HER, PPO) in two multi-goal environments, achieving a 100% success rate for almost all tasks with DDPG+HER and a sparse reward function after a few hundred epochs. The method of DDPG and HER seems promising in solving the sample inefficiency problem.

Observation Space in Deep Reinforcement Learning. Recent research has examined the impact of environment design and observation space on deep reinforcement learning algorithms. Reda et al. (2020) found that joint angles augmented with joint Cartesian positions could enhance learning speed for complex tasks, while adding binary contact information to the state space may not be helpful [5].

R. Gulde, K. Nguyen, M. Tuscher, O. Riedel, and A. Verl are with the Institute of Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, 70174 Stuttgart, Germany (phone: 0049-711-685-82505; fax: 0049-711-685-72505; e-mail: ralf.gulde@isw.uni-stuttgart.de).
The Research is supported by the Graduate School of Excellence Advanced Manufacturing Engineering and by the German Research Foundation (DFG).

Similarly, Kim et al. (2021) discovered that the observation spaces they tested were adequate in most cases, except for the raw information-only set [2]. Kozlov and Myasnikov (2022) found that increasing the amount of information in the observation space does not always improve performance [6]. These studies emphasize the importance of selecting an appropriate observation space for optimal performance in RL processes.

Task-based robotics. The field of task-based robotics has been widely studied for its potential to address general problems of intelligent systems. For example, Gienger et al. (2008) proposed a method to solve the coupled problem of grasp choice and reaching motion by learning object-dependent task maps [7]. Task space controllers have been used to enable robots to safely and efficiently explore their environment (Dries et al., 2017 [8]). Recent studies have incorporated robotics task space into deep reinforcement learning algorithms to increase sample efficiency. Bellegarda and Byl (2019) used the Proximal Policy Optimization (PPO) algorithm to train the quadrupedal spider-like robot Robosiman for the complex locomotion task of skating [9]. They found that the lack of prior knowledge of the system during training in current reinforcement learning for robotics leads to high sample inefficiency. To address this challenge, they suggest incorporating commonly available control techniques, such as forward and inverse kinematics, to reduce the agent's reliance on learning from scratch. Their approach, "Full system trained in Cartesian space with IK" (FS in CS), has shown higher sample efficiency and better performance than other systems tested. Similarly, Duan et al. (2021) used the task space approach with PPO to train the bipedal robot Cassie to walk [10]. Their work focuses more on defining the RL reward, and their experiment results demonstrate increased sample efficiency. These studies highlight the importance of incorporating domain knowledge and using an appropriate observation space in the RL training of robots for complex tasks.

III. Background

A. Task Space Inverse Kinematics

We describe robot configurations as (differentiable) task mappings $\phi : \mathbb{R}^n \to \mathbb{R}^d$, $\phi : q \mapsto y$, mapping a joint configuration $q \in \mathbb{R}^n$ to an arbitrary feature $y \in \mathbb{R}^d$ [8]. The position and orientation of the end-effector are standard task maps. However, this formulation of robot kinematics allows us to incorporate more complex task maps, such as distance, collision, and relative orientation. Task Space IK can then be defined as an optimization problem

$$q^* = \arg\min_q \|\phi(q) - y^*\|^2 \tag{1}$$

where ϕ incorporates all task maps and y^* is the corresponding desired task vector. Using its local linearization at q_0, the optimum of Eq. 1 is given by

$$q^* = q_0 + J^T(JJ^T + \epsilon\mathbf{I})(y^* - y_0)$$

where $J^T(JJ^T + \epsilon\mathbf{I})$ is the singularity robust regularized pseudo-inverse using the Jacobian J. The Jacobian for each task map ϕ w.r.t. q is retrieved by applying the forward chain rule

$$\frac{df}{dx} = \frac{\partial f}{\partial x} + \sum_{\substack{g \in \pi(f) \\ g \neq x}} \frac{\partial f}{\partial g}\frac{dg}{dx}$$

to each individual derivative within the kinematic tree.

B. Reinforcement Learning and Policy Optimization

Further, we define the Reinforcement Learning (RL) problem and introduce our notation throughout the paper. This paper regards a finite-horizon, discounted Markov Decision Process (MDP). At each timestep t, the RL-agent observes the current state $s_t \in S$, chooses an action $a_t \in A$, and then receives a reward $r_{t+1} \in \mathbb{R}$. After that, the resulting state s_{t+1} will be observed, determined by the unknown dynamics of the environment $p(s_{t+1}|a_t, s_t)$. An episode has a pre-defined length of T time steps. The goal of the agent is to find a parameter θ of a policy $\pi_\theta(a|s)$ that maximizes the expected cumulated reward J over a trajectory

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \pi(a_t|s_t) \sum_{k=t}^{T} \gamma^{k-t} r_{k+1}\right], \tag{2}$$

where $\gamma \in [0, 1]$ is the discount factor.

RL methods solve an MDP by interacting with the system and accumulating the obtained reward. We consider several model-free policy gradient algorithms with open source implementations that frequently appear in the literature, e.g., Soft Actor-Critic approaches [11], Deep Deterministic Policy Gradient (DDPG) [12], and Proximal Policy Optimization (PPO) [13].

C. Deep Deterministic Policy Gradient (DDPG)

Our method combines demonstrations with one such method: Deep Deterministic Policy Gradients (DDPG) [14]. DDPG is an off-policy model-free reinforcement learning algorithm for continuous control that can utilize large function approximators such as neural networks. DDPG is an actor-critic method that bridges the gap between policy gradient methods and value approximation methods for RL. At a high level, DDPG learns an action-value function (critic) by minimizing the Bellman error while simultaneously learning a policy (actor) by directly maximizing the estimated action-value function with respect to the parameters of the policy.

Concretely, DDPG maintains an actor function$\pi(s)$ with parameters θ_π, a critic function $Q(s, a)$ with parameters θ_Q,and a replay buffer R as a set of tuples (s_t, a_t, r_t, s_{t+1}) for each transition experienced. DDPG alternates between running the policy to collect experience and updating the parameters. Training rollouts are collected with extra noise for exploration: $a_t = \pi(s) + \mathcal{N}$, where $\mathcal{N}$ is a noise process. During each training step, DDPG samples a mini-batch consisting of $\mathcal{N}$ tuples from R to update the actor and

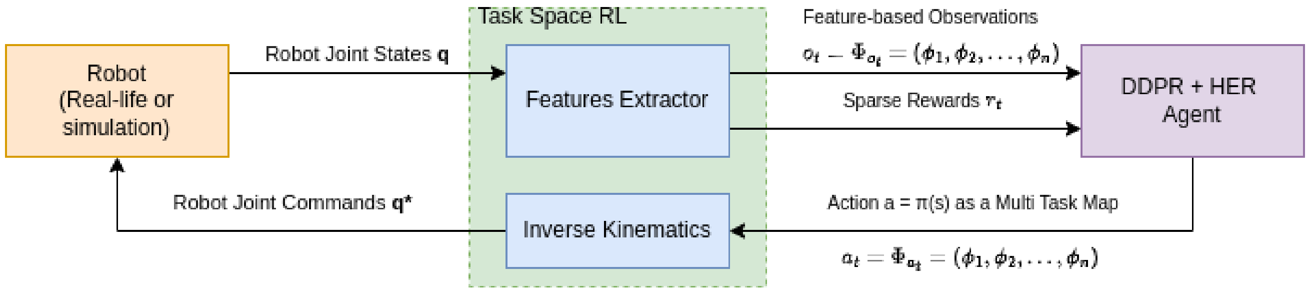
Task Space RL
Robot (Real-life or simulation)
Robot Joint States **q**
Features Extractor
Feature-based Observations
$o_t = \Phi_{o_t} = (\phi_1, \phi_2, \ldots, \phi_n)$
Sparse Rewards r_t
DDPR + HER Agent
Robot Joint Commands **q***
Inverse Kinematics
Action a = π(s) as a Multi Task Map
$a_t = \Phi_{a_t} = (\phi_1, \phi_2, \ldots, \phi_n)$

Fig. 1: Overall framework architecture.

critic networks. DDPG minimizes the following loss L w.r.t. θ_Q to update the critic:

$$y_i = r_i + \gamma Q(s_i + 1), \pi(s_i + 1)) \tag{3}$$

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i, |\theta_Q))^2 \tag{4}$$

The actor parameters θ_π are updated using the policy gradient:

$$\nabla_{\theta_\pi} J = \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta_Q)|_{s=s_i, a=\pi(s)} \nabla_{\theta_\pi} \pi(s|\theta_\pi)|_{s_i} \tag{5}$$

To stabilize learning, the Q value in equation 5 is usually computed using a separate network (called the target network) whose weights are an exponential average over time of the critic network. This results in smoother target values. Note that DDPG is a natural fit for using demonstrations. Since DDPG can be trained off-policy, we can use demonstration data as off-policy training data. We also use the action-value function $Q(s, a)$ learned by DDPG to use demonstrations better.

D. Multi-Goal RL

Instead of the standard RL setting, we train agents with parametrized goals, which lead to more general policies [15] and have recently been shown to make learning with sparse rewards easier [16]. Goals describe the task we expect the agent to perform in the given episode; in our case, they specify the desired positions of all objects. We sample the goal g_t at the beginning of every episode. The function approximators, here π and Q, take the current goal as an additional input.

E. Hindsight Experience Replay (HER)

We use Hindsight Experience Replay (HER) [17] to handle varying task instances and parametrized goals. The critical insight of HER is that even in failed rollouts where no reward was obtained, the agent can transform them into successful ones by assuming that a state it saw in the rollout was the actual goal. HER can be used with any off-policyRL algorithm, assuming that for every state, we can find a goal corresponding to this state (i.e., a goal that leads to a positive reward in this state). For every episode the agent experiences, we store it in the replay buffer twice: once with the original goal pursued in the episode and once with the goal corresponding to the final state achieved in the episode, as if the agent intended to reach this state from the very beginning.

IV. Feature-Space Observations for RL

A. Problem Formulation

We consider robot manipulation tasks that can be formulated as moving an object to a goal position. To perform complex manipulation tasks, the robot can employ tools such as grippers and screwdrivers. This family of problems includes common household and industrial manipulation tasks, such as assembling and disassembling parts together, inserting and ejecting objects, and movement in high-friction domains. We assume, that a sequence of robot actions exists to fulfill the manipulation task. However, we make no particular assumptions on the dynamics experienced throughout the task execution, and in particular, do not restrict contacts and friction.

Let ϕ_t and x_t denote the state of the robot in configuration space (or joint space) and task space, respectively, at time t, and let u_t denote the control command (target vector of Task Maps) applied at that time. Given a goal state in task space x_g, an initial robot state ϕ_0, and a time horizon T, our manipulation problem is formulated as

$$\min_{u_0,\ldots,u_T} \mathcal{L}(x_t = f_{FK}(\phi_t), s_g), \tag{6}$$

$$s_{t+1} = f(\phi_t, u_t), \; t \in 0, ..., T. \tag{7}$$

where f is the (unknown) system dynamics,f_{FK} is the forward kinematics function.

B. MDP Architecture

To provide a concrete example, we consider the robot pick and place task for illustration purposes. Concerning the introduced notation for Task-Space Robotics, we define observations, actions and rewards of the Markov Decision Process (MDP) as follows:

Observation. The Agents observation o_t is described by a Multi Task Map Φ (multiple Task Maps aggregated to a vector) $o_t = \Phi_{o_t} = (\phi_1, \phi_2, ..., \phi_n) \in \mathcal{S}$. For a robotic pick and place task the observations space can be defined using the following Task Maps $\Phi = (\phi_1, \phi_2, ..., \phi_7)$:

- ϕ_1: actual object position $x_{act,obj,pos}$,
- ϕ_2: goal object position $x_{goal,obj,pos}$,

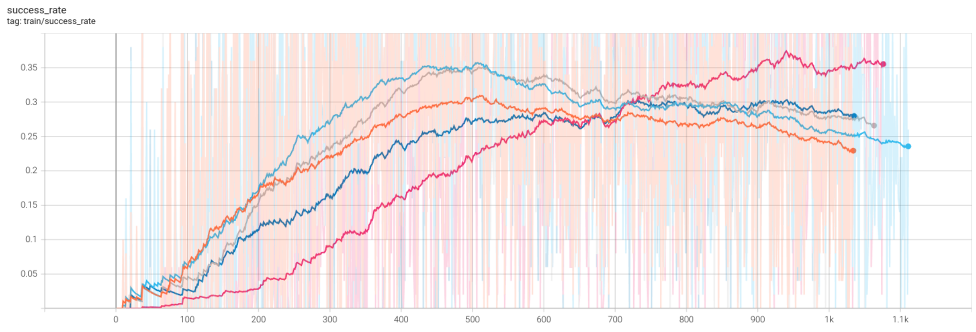

Fig. 2: The observation space configurations affect the success rate during RL training. Graph notation: *Orange*: baseline, *Red*: Simple Observation, *Dark blue*: Feature Set 0, *Light Blue*: Feature Set 1, *Grey*: Feature Set 2

- ϕ_3: end-effector position $x_{act,EE,pos}$,
- ϕ_4: end-effector orientation in quaternion $x_{act,obj,ori}$,
- ϕ_5: position difference $x_{act,obj,pos}$ and $x_{goal,obj,pos}$,
- ϕ_6: position difference $x_{act,EE,pos}$ and $x_{goal,obj,pos}$.

Action. The agents action a_t is defined by a Multi Task Map $a_t = \Phi_{a_t} = (\phi_1, \phi_2, ..., \phi_n)$. In the robotic pick and place scenario, the actions are defined by $a_t = \Phi_{a_t} = (\phi_1, \phi_2, ..., \phi_4)$:

- ϕ_1: target end-effector position $x_{targ,EE,pos}$,
- ϕ_2: scalar product of the x-axes of robot base frame and end-effector frame $< \mathcal{F}_{base,x}, \mathcal{F}_{EE,x} >$,
- ϕ_4: scalar product of the y-axes of robot base frame and end-effector frame $< \mathcal{F}_{base,y}, \mathcal{F}_{EE,y} >$,
- ϕ_3: scalar product of the z-axes of robot base frame and end-effector frame $< \mathcal{F}_{base,z}, \mathcal{F}_{EE,z} >$,
- ϕ_4: target gripper command $x_{targ,grip,pos}$.

Reward. We use a sparse reward signal to reinforce the agent:

$$r_{t+1} = \begin{cases} 1, & \mathcal{L}(\Phi_{act}, \Phi_{des}) < \epsilon \\ 0, & otherwise \end{cases}, \tag{8}$$

As stated in formula 8, we compute the scalar reward signal by computing a loss functional $\mathcal{L}$ using an actual and a desired Task Map. In case the threshold condition $\mathcal{L}(\Phi_{act}, \Phi_{des}) < \epsilon$ is fulfilled, the agent receives a non-zero reward. In the ick and place scenario, the threshold condition can be formulated as $\mathcal{L}(\phi_{_{goal,obj,pos}}, \phi_{x_{act,obj,pos}}) < \epsilon$

C. Feature-based observation RL framework

As illustrated in Figure 1, our proposed framework aims to integrate the previously formulated Markov Decision Process (MDP) into the reinforcement learning algorithms. The framework comprises three fundamental components: the robot environment, a task-space calculator, and reinforcement learning agents. The robot environment can be instantiated as a physical or simulated system, which can be accurately emulated using simulators such as MuJoCo, PhysX, or Gazebo. The task-space calculator encompasses two essential elements: a feature extractor and an inverse kinematic solver. Inspired by the work of Saeed et al. [4], we adopt a combined Deep Deterministic Policy Gradient (DDPG) and Hindsight Experience Replay (HER) approach within our framework, as it has been shown to achieve high sample efficiency in robot learning.

The robot environment provides the current robot joint state q to the feature extractor component of the task-space calculator. The feature extractor processes the joint state and augments the observation space with a set of robotics features. Additionally, the rewards are computed and provided to the reinforcement learning agents. Based on this information, the DDPR+HER agent learns an optimal policy and selects an appropriate action. The action, a set of robotics features represented as a Multi-Task Map, is then processed by the task-space calculator to obtain the inverse kinematic and transmits the desired joint state to the robot environment for execution.

V. Experiment

The experiment focuses on the Reacher robotics task, wherein a robot aims to reach a randomly determined position within a 3D environment. To address this task, we implement the framework illustrated in Figure 1. In light of its reliability, efficiency, and versatility in robotics physics simulation, we select the widely recognized Mujoco [18] as our simulator.

In Figure 3, we simulate the Universal Robot 5 equipped with a suction gripper as its end effector. In each episode, a target, denoted by a red dot, is randomly designated, and the robot's objective is to navigate toward this target. As Mujoco solely provides information regarding the robot's joint state, we employ the Robotics AI (RAI) []toussaintNewtonMethodsKorder2014b library to extract task space information and enhance the observation spaces by incorporating this additional information. Moreover, the action space operates in the task space domain. Subsequently, after the agent selects an action, RAI calculates the inverse kinematics and communicates the resultant instructions back to Mujoco for execution. Notably, the observation space differs across each experiment. Regardless, we employ the DDPG+HER (Deep Deterministic Policy Gradient with Hindsight Experience Replay) algorithm as our Reinforcement Learning agent for all experiments. The actor and critic components of the agent are implemented as feed-forward neural networks with hidden layer dimensions of [256, 256].

TABLE I: Observation space configurations

Configurations	Dimension	Features ϕ
Baseline	6	• End-effector 3D position • End-effector 3D velocity
Simple features	3	• 3D position difference between end-effector and target. • The velocity of the 3D position difference between the end-effector and target.
Feature set 0	9	• End-effector 3D position • Target 3D position • 3D position difference between end-effector and target.
Feature set 1	15	• Feature set 0. • End-effector 3D velocity • The velocity of the 3D position difference between the end-effector and target.
Feature set 2 (full state)	23	• Feature set 1. • End-effector quaternion. • Angular difference between target and end-effector.

The learning rate for both networks is set at 0.001, and network updates occur after every two episodes.

As the baseline for our study, we utilize the Reacher-v3 Gymnasium environment developed by the Farama Foundation [26]. This environment's observation space comprises the 3D position and 3D velocity of the robot's end effector, while the action space is limited to the 3D position of the end effector.

We conduct a benchmark analysis of various observation space configurations throughout the experiment. These configurations encompass distinct robotic features, such as end effector position, velocity, and quaternion representation. The specific details of these configurations can be found in Table I.

VI. Result and Discussion

Figure 2 illustrates the impact of observation space configurations on the learning process of Reinforcement Learning algorithms. The graph provides clear evidence of the critical importance of carefully selecting the observation space to achieve sample efficiency and robustness in RL.

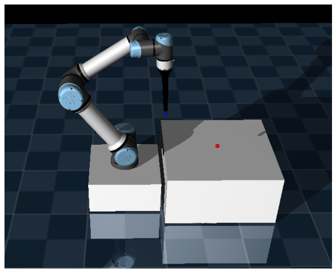
Fig. 3: Universal Robot 5e Reacher task

Our feature-based approaches, namely feature set 1 and feature set 2, have demonstrated superior performance compared to the standard baseline from the Gymnasium Reach environment. These approaches exhibit faster convergence and higher success rates by incorporating additional and enriched feature-based information, particularly in velocity and speed.

The Simple Features configuration achieves the highest final success rate despite its slower initial convergence due to limited information. This observation suggests that RL performance can improve significantly even with relatively low dimensions but highly informative feature sets.

These findings underscore the significance of selecting observation space configurations that provide relevant and enriched feature-based information, enabling RL algorithms to converge more efficiently and achieve higher success rates.

VII. Conclusion

In conclusion, Reinforcement Learning (RL) has become a popular approach for intelligent robotic systems, but it still faces challenges related to sample inefficiency and lack of generalization. Our proposed approach utilizes readily available control techniques and enhances the observation space with task-space feature sets to reduce the need for the agent to learn everything from scratch. Our approach also helps eliminate the need for the robot to re-learn inverse kinematics, resulting in increased sample efficiency. Future work will focus on applying our approach to various robotics applications to evaluate their effectiveness and robustness.

References

[1] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, Sep. 2021.

[2] J. T. Kim and S. Ha, "Observation Space Matters: Benchmark and Optimization Algorithm," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 1527–1534.

[3] X. Ni, X. He, and T. Matsumaru, "Training a Robotic Arm Movement with Deep Reinforcement Learning," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2021, pp. 595–600.

[4] M. Saeed, M. Nagdi, B. Rosman, and H. H. S. M. Ali, "Deep Reinforcement Learning for Robotic Hand Manipulation," in *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, Feb. 2021, pp. 1–5.

[5] D. Reda, T. Tao, and M. van de Panne, "Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning," in *Motion, Interaction and Games*, Oct. 2020, pp. 1–10.

[6] D. Kozlov and V. Myasnikov, “The impact of a set of environmental observations in the problem of acquiring movement skills in three-dimensional space using reinforcement learning algorithms,” in *2022 VIII International Conference on Information Technology and Nanotechnology (ITNT)*, May 2022, pp. 1–5.

[7] M. Gienger, M. Toussaint, and C. Goerick, “Task maps in humanoid robot manipulation,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nice: IEEE, Sep. 2008, pp. 2758–2764.

[8] D. Dries, P. Englert, and M. Toussaint, “Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore: IEEE, May 2017, pp. 902–907.

[9] G. Bellegarda and K. Byl, “Training in Task Space to Speed Up and Guide Reinforcement Learning,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 2693–2699.

[10] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, “Learning Task Space Actions for Bipedal Locomotion,” May 2021.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” *arXiv preprint arXiv:1801.01290*, 2018.

[12] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *ICML*, 2014.

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

[14] 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.

[15] T. Schaul, D. Horgan, K. Gregor, and D. Silver, “Universal value function approximators,” in *International conference on machine learning*, 2015, pp. 1312–1320.

[16] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[17] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, “Hindsight experience replay,” in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[18] 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.

[19] M. Toussaint, “Newton methods for k-order Markov Constrained Motion Problems,” Jul. 2014.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016, pp. 1928–1937.

[21] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, “Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research,” *arXiv:1802.09464 [cs]*, Mar. 2018.

[22] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Overcoming Exploration in Reinforcement Learning with Demonstrations,” *arXiv:1709.10089 [cs]*, Feb. 2018.

[23] G. Zuo, Q. Zhao, J. Lu, and J. Li, “Efficient hindsight reinforcement learning using demonstrations for robotic tasks with sparse rewards,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 1, p. 172988141989834, Jan. 2020.

[24] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, “Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo,” *arXiv preprint arXiv:1608.05742*, 2016.

[25] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum, “Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning,” in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, Jun. 2018.

[26] Gymnasium-Robotics Documentation. [Online]. Available: https://robotics.farama.org/index.html

[27] R. Nagpal, A. U. Krishnan, and H. Yu. Reward Engineering for Object Pick and Place Training. [Online]. Available: http://arxiv.org/abs/2001.03792

[28] M. Toussaint. Newton methods for k-order Markov Constrained Motion Problems. [Online]. Available: http://arxiv.org/abs/1407.0414

[29] ——, “RAI bare code.” [Online]. Available: https://github.com/MarcToussaint/rai

[30] G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, “Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 10 364–10 370.

[31] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-End Training of Deep Visuomotor Policies,” Apr. 2016.

[32] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, “Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review,” *Robotics*, vol. 10, no. 1, p. 22, Jan. 2021.

[33] J. Peters and S. Schaal, “Learning Operational Space Control,” in *Robotics: Science and Systems II*. Robotics: Science and Systems Foundation, Aug. 2006.

[34] O. Petrovic, L. Schäper, S. Roggendorf, S. Storms, and C. Brecher, “Sim2Real Deep Reinforcement Learning of Compliance-based Robotic Assembly Operations,” in *2022 26th International Conference on Methods and Models in Automation and Robotics (MMAR)*, Aug. 2022, pp. 300–305.

[35] L. Smith, I. Kostrikov, and S. Levine, “A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning,” Aug. 2022.