Proceedings in Manufacturing Systems, Volume 15, Issue 3, 2020, 135-146 ISSN 2067-9238

REINFORCEMENT LEARNING FOR ASSEMBLY ROBOTS: A REVIEW

Liliana STAN[1,*], Adrian Florin NICOLESCU[2], Cristina PUPĂZĂ[2]

[1)] PhD Student, Robots and Manufacturing Systems Department, University "Politehnica" of Bucharest, Romania
[2)] Prof., PhD, Robots and Manufacturing Systems Department, University "Politehnica" of Bucharest, Romania

Abstract: *This paper provides a comprehensive introduction to Reinforcement Learning (RL), summarizes recent developments that showed remarkable success, and discusses their potential implications for the field of robotics. RL is a promising approach to develop hard-to-engineer adaptive solutions for complex and diverse robotic tasks. In this paper RL core elements are reviewed, existing frameworks are presented, and main issues that are limiting the application of RL for real-world robotics, such as sample inefficiency, transfer learning, generalization, and reproducibility are discussed. Multiple research efforts are currently being directed towards closing the sim-to-real gap and accomplish more efficient policy transfers methods, making the agents/robots learn much faster and more efficiently. The focus of this work is to itemize the various approaches and algorithms that center around the application of RL in robotics. Finally, an overview of the current state-of-the-art RL methods is presented, along with the potential challenges, future possibilities, and potential development directions.*

Key words: *reinforcement learning, robot arm, robotic vision, manipulation tasks.*

1. INTRODUCTION

Reinforcement Learning (RL) has attracted a lot of attention in recent years with breakthroughs in multiple domains, including robotics.

Industry 4.0 is characterized by modularity, inter-operability, and real-time capabilities. To address the custom manufacturing demands, RL is a key catalyst for turning an industrial robot which is designed for a fixed and repetitive task into a 'smart manipulator', having the capability to learn and perform a desired task without any explicit task-specific controller. For that to happen, present-day controllers must be augmented with learning-based solutions for motion planning, trajectory tracking control, collision avoidance, force control, and robotic vision.

Recent studies show exciting progress for RL in robotics. Deep Learning (DL) has been applied successfully to many important problem areas, including computer vision, robotics and RL. The current challenges entail solutions for scaling up to complex tasks for robots, designing robust policy representations, and optimizing the computing time.

Precise, collision-free, trajectory tracking with optimal control has been an active research area where progress is needed to a great extent [1, 2, 3]. A vast amount of research emerged for tracking applications using manipulator arms, from simple tasks such as pick-and-place [4], peg-in-hole [5], cloth folding [6], to more complex tasks, such as tracking a trajectory on an irregular surface in a multi-robot system while avoiding self-collisions, and cooperative handling operations [7].

[*] Corresponding author: 313 Splaiul Independenței, Bucharest, sector 6, Romania,
Tel.: +40720579271
E-mail addresses: *elena_liliana.stan@stud.fiir.upb.ro* (L. Stan)

Currently there are several active RL subfields of research that enable breakthroughs in robotics with regards to path planning, trajectory tracking control, optimal control, robot impedance and force control, compliance control, and robot vision such as object tracking, pose estimation, and pattern recognition [8]. Promising RL tasks have also been presented addressing the security and energy consumption of robots. In [9], Bozkurt et al. addressed the problem of security-aware robotic motion planning and Yin el al. proposed a machine learning-based approach for energy-efficient trajectory planning [10].

Despite these very promising results, applications of RL to industrial robotics are currently rather limited, and this can be attributed to the laborious efforts to setting-up the learning framework based on recent research, and to the lack of experimental evaluation of RL-based methods.

To address the problem of reproducibility of deep RL research, Rupam et al. developed a learning task with an UR5 robotic arm and discussed the key elements of real-world task setups [11].

Robotic assembly is a manufacturing process in which a robot (equipped with tailored end-effectors) positions, mates, fits and assembles interchangeable parts or sub-assemblies in a sequential manner resulting a functional product (Fig. 1). This process requires a high degree of repeatability, reliability, flexibility, and sequence planning. Typical robot-assembly operations also require compliance and force control that provide stable contact between the manipulator and the workpiece.

Assembly sequences are predefined manually in the traditional robotic-assembly systems. In the context of smart factories, for a successfully assembled product, a RL-based assembly approach should be able to plan the correct sequences of assembly tasks, to plan individual,

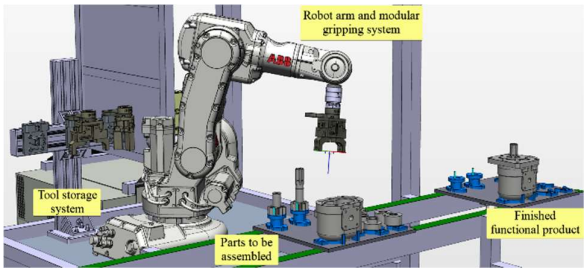

Fig. 1. A typical setup of a robotic assembly cell.

collision-free movements, to calculate the required forces and torques, and to estimate the pose of assembly parts. Since different end-effectors (modular grippers, electric screws, glue applicators, etc.) are needed in an assembly process, knowing the correct sequence of assembly tasks, the RL model should also make sure that the robot equips the proper end-effectors. All these requirements raise major difficulties for researchers, as robotic assembly continues to be one of the most challenging problems in the field of robotics research. Assembly tasks such as pick-and-place, peg-in-hole, bolt screwing, slide-in-groove, stacking, pushing are well-covered in the literature.

The research area for assembly sequence planning (ASP) has been active this past decade and solutions have been proposed based on various algorithms [12–15] to improve assembly efficiency, reduce production costs, and shorten development cycles. In [16] the influence of assembly predicates on optimal assembly sequence generation, in terms of search space, computational time and possibility of resulting practically not feasible assembly sequences is discussed. Assembly predicates are used to test the possibility of assembling components in the defined sequence in the physical environment and they are defined based on the assembly connections, part geometries, and accessibility to perform the assembly operations. Recently, Watanabe and Inada [17] proposed a computational algorithm for searching the efficient assembly sequence and work assignment, in the context of future smart factories. A RL framework is used to search the assembly sequence through trial and error, and neural networks methods are used to transfer the past learning results to the new search to improve the search performance.

Learning from demonstration (LfD) is the paradigm in which agents acquire new skills by learning to imitate an expert [18, 19]. LfD enables robots to move away from repeating simple prespecified behaviors in constrained environments and toward learning to take optimal actions in unstructured ones without placing a significant burden on the operator [20]. Demonstrations can be performed by means of kinesthetic teaching, teleoperation, or passive observation.

This paper provides a summary of some of the main components for applying RL in robotics and includes state-of-the art RL methods and algorithms. The main contribution of this work is a better understanding that the design of appropriate policy representations is essential for RL methods to be successfully applied to real-world robots.

This paper is organized as follows: section II describes key RL concepts and challenges. Next, section III focuses on RL models and algorithms specific to 6 DOF robot manipulators. Section IV presents recent developments in the field of robot vision. In Section V popular RL simulation frameworks and environments for robotics are presented. Section VI concludes the review.

2. RL – KEY CONCEPTS AND TERMINOLOGY

Reinforcement learning (RL) is prominently used for sequential decision-making. The RL problem can be formalized as an agent that learns an optimal policy by trial and error, to make decisions in an environment to optimize a given notion of cumulative rewards. After taking an action in a specific state, the agent (in the literature often referred to as a *controller*) receives a scalar reward from the environment, which provides an indication of the quality of that action. The function that indicates the action to take in a certain state is called a *policy*. By following a given policy and processing the rewards, the agent can build estimates of the accumulated reward (also called the *return*). The function representing the estimated return is known as the *value function*.

Over the course of time, several types of RL algorithms and methods have been introduced and they can be divided into three groups: actor-only (picks actions greedily based on state values), critic-only (uses the policy function to pick actions) and actor-critic (uses both value and policy functions).

The actor-critic method works as follows: at time step *t* the actor senses the current system state and applies an action based on policy (Fig. 2). This leads to a new system state and a numerical reward. Using this, the Temporal difference (TD) error is calculated. Using this error, the critic updates its estimate of the optimal value function. For continuous state and action spaces, such as in the case of a robot manipulator arm, the actor and critic need to be approximated. In contrast to critic-only methods, actor-critic methods usually have good convergence properties, [21].

Deep learning (DL), a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. A deep neural network is characterized by a succession of multiple processing layers [22]. Deep Reinforcement Learning (DRL) combines artificial neural networks with a RL architecture that enables software-defined agents to learn the best actions possible in a virtual environment in order to attain their goals.

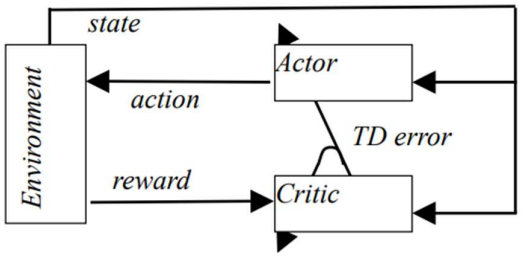

Fig. 2. The actor-critic agent's architecture [23].

A Markov Decision Process (MDP) is a tuple ($\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{T}$, $\boldsymbol{R}$, $\boldsymbol{\gamma}$), where $\boldsymbol{S}$ is the *state* space, $\boldsymbol{A}$ is a finite set of *actions*, $\boldsymbol{T}$ is the state *transition probability distribution*, $\boldsymbol{R}$ is the *reward* function, and $\boldsymbol{\gamma}$ is a *discount factor* $\boldsymbol{\gamma} \in [0, 1]$. The agent learns a quality-function (Q-learning) denoting the sum of rewards from state *s* onward if action *a* is taken, and as such the agent can choose what to do in state *s* by finding the action with the highest Q-value. In the policy search, the agent learns a policy that maps directly from states to actions. More state-of-the-art methods and approaches will be presented in Section 3.5.

In a *model-based* RL approach, the agent uses a transition model of the environment to help interpret the reward signals and to make decision about how to act (the agent learns a utility function). In contrast, in a *model-free* RL the agent neither knows nor learns a transition model for the environment. Instead, it learns a more direct representation of how to behave, either through action-utility learning (quality-function) or through policy search (policy mapping).

Ebert et al. [24] proposed a self-supervised model-based approach for a robotic manipulation task that generalizes effectively to never-before-seen tasks and objects (visual MPC). Recently, a novel approach to alleviate the data inefficiency of model-free RL by implicitly leveraging model priors, called Model-Based Baseline was proposed in [25].

On-Policy vs. Off-Policy: every RL algorithm must follow a certain policy to decide which actions to perform at each state. Algorithms that concern about the policy which yielded past state-action decisions are referred to as on-policy algorithms, while those ignoring it are known as off-policy. Q-Learning is an off-policy RL algorithm, considered as one of the very basic ones. In its most simplified form, it uses a table to store all Q-Values of all possible state-action pairs possible. SARSA is a slight modification of Q-Learning in order to make it an on-policy algorithm. SARSA updates its Q-values using the Q-value of the next state s′ and the current policy's action a″. Wang et al. proposed the backward Q-learning based SARSA algorithm which could enhance the learning speed and improve action quality and total performance [26].

Exploration vs Exploitation: the agent can try many different actions in many different states in order to try and learn all available possibilities and find the path which will maximize its overall reward (explores the environment), or the agent can use the information learned (exploit) to maximize the rewards it receives. Exploration gives more knowledge about the environment (leading to better future decisions), while exploitation chooses the best action to take given the current information (advancing to the current most promising direction) [27]. The best strategy involves sacrificing short-term rewards for more reward in the future, meaning that a balance between exploration and exploitation is needed, known as the *exploitation-exploration trade-off* [28], which is fundamental to many RL algorithms.

Imitation Learning is considered as an alternative to RL to solve sequential decision-making problems and aims to train a policy to mimic the behavior of an expert, given only the demonstrations from that expert. It is also closely related to the LfD approach. There are currently two main paradigms: *behavior cloning* (where a policy is trained only as a supervised-learning task) and *inverse RL* (the goal is to recover a reward function of the MDP that can explain the behavior of the expert demonstrator).

2.1. Challenges that RL faces in robotics

RL is applied in robotics, with techniques striving to reduce the required number of interactions with the real world. Such techniques tend to exploit models, be it estimating models and using them to plan, or training policies that are robust with respect to different possible model parameters. The high number of degrees of-freedom of modern robots leads to large dimensional state spaces, which are difficult to be learned: example demonstrations must often be provided to initialize the policy and mitigate safety concerns during training. Bellman coined the term “Curse of Dimensionality” in 1957 [30], when he explored optimal control in discrete high-dimensional spaces and faced an exponential explosion of states and actions. As the number of dimensions grows, exponentially more data and computation are needed to cover the complete state-action space.

Some remaining barriers to the adoption of RL and DRL in robotics include the necessity for *large training data* and *long training times.* Banko and Brill [18] argued that the performance of learners can benefit significantly from much larger training sets (results observed for a natural language classification task). Generating training data on physical systems can be relatively time consuming, expensive, and sometimes unsafe. Moreover, in a simulated environment, it is possible to use learning algorithms running in parallel computation, to learn in a few hours from millions of trials. But in a real environment, it might take years to run these same trials.

There is also the problem of *under-modeling and uncertainty*. Simulation with accurate models could potentially be used to offset the cost of real-world interaction. In an ideal setting, this approach would allow to learn the behavior in simulation and subsequently transfer it to the real robot. Unfortunately, creating a sufficiently accurate model of the robot and its environment is challenging and often requires extensive data samples (Fig. 3). As small model errors due to this under-

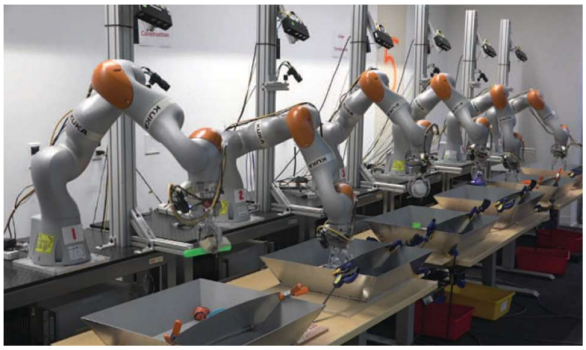

Fig. 3. Large-scale data collection setup consisting of 8 Kuka IIWA robots, used to evaluate transfer learning after another setup was used to collect a dataset consisting of over 800,000 grasp attempts [29].

modeling accumulates, the simulated robot can quickly diverge from the real-world system.

The solutions to these problems came from an active area of research, which enables the transfer of what has been learned in simulation to a real robot in the real world, known as *sim-to-real transfer.*

Another way to reduce the number of real-world samples required for learning is to reuse information from previous learning episodes on other tasks, rather than starting from scratch. This falls under the umbrella of *meta-learning* or *transfer learning.*

The reality gap represents all discrepancies between physics simulators and the real world that make transferring behaviors from simulation challenging. To learn a policy that transfers well, adding noise to the model during training, can make the policy more robust (*data augmentation*) or, policies can be trained that will work with a variety of models by sampling different parameters in the simulations (*domain randomization*).

Mahmood et al. benchmarked RL algorithms on a robot arm and found that state-of-the-art learning algorithms are highly sensitive to their hyper-parameters and their relative ordering does not transfer across tasks, indicating the necessity of re-tuning them for each task for best performance [30].

Barriers addressing the *safety concern*: applying RL in robotics demands safe exploration which becomes a key issue of the learning process. While learning, the dynamics of a robot can change due to many external factors ranging from temperature to wear thereby the learning process may never fully converge. A possible solution is to learn low-level control policies for motion planning using soft robots [29] (manipulators with joint torque control or flexible joints), which are considerably safer to work with due to their compliant nature.

The main robotic assembly challenge is the requirement of a high degree of repeatability, reliability, and flexibility. Two common approaches for RL practitioners for faster learning and better performance are the *Hierarchical Task Decompositions* and the *Skill Reusability* which ensure that the primary robotic task is divided into smaller and more tractable problems in order for the robot to learn skill policies for performing the lowest level of tasks and then use these skills as an action basis to perform the next level of tasks. *Transfer learning* tries to use experience from one set of tasks on a new task. Finding an abstract representation and generalizing across objects is a major aspect of learning to manipulate objects and adapt accordingly.

Despite its impressive successes, RL and DRL still face significant obstacles, but it is, nonetheless, a very active area of research. As hardware improved (leading to specialized hardware e.g., GPU, TPU, or FPGA), and as RL and neural networks began to become more practical, they were increasingly found to be effective with real robotics applications.

Another important development in the field of RL has been indirectly borrowed from vast successes of deep convolutional neural networks (CNN) in image feature extraction. Work in RL has been also accelerated by the availability of open-source simulation environments for developing and testing learning agents, which will be further addressed in Section V.

3. RL IN ROBOTICS

The main motivation for using RL to teach robots new skills is that it offers three previously missing abilities: to learn new tasks that otherwise cannot be directly programmed, to learn to achieve optimization goals of difficult problems that have no analytic formulation by using only a known cost function (e.g., minimize the used energy for performing a task), to learn to adapt a skill to a new, previously unseen version of a task.

The predominant approach to perception, planning, and control in robotics is to use approximate models of the physics underlying a robot, its sensors, and its interactions with the environment. These model-based techniques often capture properties such as the mass, momentum, shape, and surface friction of objects, and use these to generate controls that change the environment in a desirable way. In contrast, physics-based models are well suited for planning and predicting the outcome of actions. To function on a real robot, they require that all relevant model parameters are known with sufficient accuracy and can be tracked over time. This requirement poses overly challenging demands on system identification and perception.

Robotic learning in the physical world is also constrained by time, by the risk of damaging the robot (or putting at risk the safety of its surroundings) and by the large data needed for training (expensive to collect). Data augmentation, transfer learning, and sim-to-real are efficient solutions to this problem. Sim-to-real is a technique often used to train in simulation before attempting to train with an actual robot, this way reducing the wear on physical equipment, and the training time.

To speed up RL, Dai et al. [31] proposed a novel method for Self-Imitation Learning (SIL), in which an on-policy RL algorithm uses episodic modified past trajectories, i.e., hindsight experiences, to update policies. The experiments show that episodic SIL can perform better than baseline on-policy algorithms, achieving comparable performance to state-of-the-art off-policy algorithms in several simulated robot control tasks. With the capability of solving sparse reward problems in continuous control settings, episodic SIL has the potential to be applied to real-world problems that have continuous action spaces, such as robot guidance and manipulation.

A new domain adaptation algorithm called MPBO (Multi-Policy Bayesian Optimization) was introduced in [32] and uses simulated kinematic parameters variation. The simulated experiments showed that introducing variations in kinematics during training in simulation can benefit policy transfer. Hameed et al. [33] introduced a novel combination of scheduling control on a flexible robot manufacturing cell with curiosity based RL. In [34], the authors have analyzed how multi-agent reinforcement learning can bridge the gap to reality in distributed multi-robot systems where the operation of the different robots is not necessarily homogeneous.

One of the foremost requirements for manufacturing robots is precision, therefore, the RL model must provide a precise reference tracking. Extensive efforts have been made to improve the generalization ability of RL methods in robotics via domain randomization and data augmentation.

Recently, [35] proposed SOft Data Augmentation (SODA), a method that stabilizes training by decoupling data augmentation from policy learning. SODA is a general framework (which uses Soft Actor-Critic (SAC) as the base algorithm) for data augmentation that can be implemented on top of any standard RL algorithm. [36] proposed an adversarial RL framework, which significantly increases robot robustness over joint damage and malfunction in manipulation tasks. This enables the robot to be fault-aware of its joint working states.

To apply RL on real-world robots in safety-critical environments, it should be possible to ensure safety during and after training. During RL training, applying the trial-and-error approach to real-world robots operating in safety-critical environment may lead to collisions. To address this challenge, [37] proposed a Reachability-based Trajectory Safeguard (RTS), which leverages trajectory parameterization and reachability analysis to ensure safety while a policy is being learned. The method, RTS+RL, is demonstrated in simulation performing safe, real-time receding-horizon planning on three robot platforms with continuous action spaces.

3.1. Motion planning and trajectory tracking control

Control theory is still an active field of research addressing the motion planning problem by finding a solution that takes a robot from one configuration to another without colliding with an obstacle. The task of executing a sequence of actions to follow the path is called *trajectory tracking control*. A trajectory has time associated with each point on the path. A dynamic model (transition model) is needed for RL solutions and used for motion: knowing the configurations (given by the path), a function is used to compute the effects torques have on a configuration.

Motion planning can be solved via graph search using cell decomposition, using randomized motion planning algorithms (that tend to first construct a complex but feasible path and then optimize it), or using trajectory optimization (which starts with a simple but infeasible path, and then iteratively pushes it out of collision). The simplest method to solve the optimization problem and find a path is to use a method known as gradient descent.

In [3], the authors addressed the problem of solving manipulation tasks in the presence of obstacles and proposed a novel approach, called MoPA-RL (motion planner augmented RL), which combines the strengths of both motion planning and RL by augmenting the action space of an RL agent with the capabilities of a motion planner.

A path found by a search algorithm can be executed using the path as the reference trajectory for a PID controller, which constantly corrects for errors between where the robot is and where it is supposed to be, or via computed torque control, which adds a feedforward term that makes use of inverse dynamics to compute roughly what torque to send in order to make progress along the trajectory.

A middle ground between open-loop control based on inverse dynamics and closed-loop PID control is called computed torque control (the torque the model thinks is needed will be computed, but afterwards it is compensated for model inaccuracy with proportional error terms).

JAiLeR (Joint Space Control via Deep Reinforcement Learning), a deep RL-based approach to control a robot manipulator was presented in [38]. A deep neural network, trained via model-free reinforcement learning, is used to map from task space to joint space. Training in simulation showed that this simple approach can achieve accuracy comparable to that of classical techniques over a large workspace, in both simulation and reality. Advantages of this approach include automatic handling of redundancy, joint limits, and acceleration/ deceleration profiles.

An approach to learn fast robot trajectories while satisfying kinematic joint constraints was presented in [39]. In contrast to penalizing constraint violations, this approach provides explicit safety guarantees. Dual-arm robots have attracted much attention in recent years and solutions emerged addressing the challenges of motion planning, e.g. Simultaneous Dual-Arm Motion Planning [40] which is a method to achieve efficient pick-and-place performance with a dual-arm robot in order to minimize its operation time. The path planning using the rapidly exploring random tree (RRT) enabled the dual-arm robot to execute the pick-and-place work without collisions between the arms. Motion planning for multi-robots was also addressed in [41–43].

3.2. Optimal control

Optimal control unites motion planning and trajectory tracking by computing an optimal trajectory directly over control inputs. Put in simple words, the trajectory optimization problem for kinematic paths is turned into true trajectory optimization with dynamics. To capture a detailed background introduction to optimal control, the authors recommend Liberzon's book [44].

The linear quadratic regulator (LQR) finds many uses in practice because of the ease of finding the optimal policy. With LQR, the optimal value function is quadratic, and the optimal policy is linear. Variants of LQR are also often used for trajectory tracking. Another practical method called iterative LQR (ILQR), works by starting with a solution and then iteratively computing a linear approximation of the dynamics and a quadratic approximation of the cost around it, then solving the resulting LQR system to arrive at a new solution. ILQR is currently widely used at the intersection of motion planning and control [45].

Dean et al. addressed the problem of optimal control and proposed a multistage procedure, called Coarse-ID control [46]. The method estimates a model from few experimental trials, estimates the error in that model and then designs a controller (using both the model and uncertainty estimate). Kapoor et al. [47] proposed an approach to express the desired high-level robot behavior using a formal specification language known as Signal Temporal Logic (STL) as an alternative to reward/cost functions. The proposed algorithm is empirically evaluated on simulations of robotic systems such as a pick-and-place using a robotic arm.

Addressing the control stability concern, a normalizing flow control structure for robotic manipulation was recently introduced by Khader et al in [48]. Their results indicated that by inducing stable behavior, the state-action distribution can be significantly reduced without compromising the learning performance.

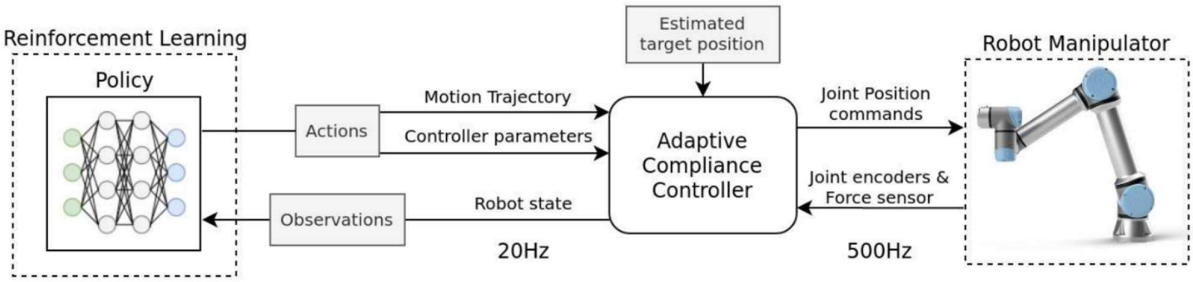

Fig. 4. A variable compliance control approach for robotic assembly proposed in [5]: the system learns a control policy that defines motion-trajectory and force-control parameters of an adaptive compliance controller to control the robot arm.

3.3. Force control, impedance, and compliance

The most popular approaches for interaction control of robot manipulators are position/force control and impedance control. Force control methods address the problem of interaction between a robot manipulator and its environment, providing direct control of the interaction through contact force feedback and a set of dynamic parameters. The position/force control uses the impedance model to generate the desired force.

The objective of the impedance control is to achieve the desired dynamic relationship between the end-effector position of the robot and the contact force, namely that the output force is as small as possible and that the position is close to its reference. The impedance model uses as input the position/velocity of the end-effector and outputs the exerted force.

Recently, Perrusquía et al. presented a model-free PID admittance controller for robot manipulators that use RL, showing that the position error is minimized without knowing the environment or the impedance parameters [49]. Various methods to actively control compliance at the end-effector have been developed, such as impedance control [51, 52] operational space control [52], hybrid force-control [54, 55], and virtual model control [55].
Impedance control is a well-established technique to control interaction forces in robotics. Active compliance is achieved through the active control of joints (position or torque) using feedback measurements of joint torques. A major benefit of active compliance is its ability to change the dynamic characteristics (e.g. stiffness and damping) in real-time.

Based on compliance control, Roveda et al. proposed an approach that consists in two main control levels: iterative friction learning compensation controller and iterative force-tracking learning controller. The control procedure has been applied to an automotive industrial assembly task [56]. A methodology for designing joint impedance controllers based on an inner torque loop and a positive velocity feedback loop was presented in [57] for safely learning contact-rich manipulation tasks on position-controlled robots.

Figure 4 shows the system architecture proposed in [5], having two control loops: the inner one is an adaptive compliance controller (a parallel position-force controller) and the outer one is an RL control policy.

In [58], the authors proposed a learning framework for position-controlled robot manipulators to solve contact-rich manipulation tasks (peg-in-hole tasks with hole-position uncertainty). An off-policy model-free RL method was used, while bootstrapping the training speed by using several transfer-learning techniques: sim-to-real and domain randomization. A learning-based force control framework combining RL techniques with traditional force control was proposed in [59].

3.4. Training models and data

A model of the world can be useful in reducing the sample complexity of model-free RL methods by doing *sim-to-real transfer*: transferring policies that work in simulation to the real world (illustrated in Fig. 5).

To learn hand-eye coordination for robotic grasping, Levine et al. trained a large convolutional neural network to predict the probability that task-space motion of the gripper will result in successful grasps [60]. Their large-scale data collection setup consisted of 14 robotic arms running over the course of two months to collect 800,000 grasp attempts.

More recent techniques have experimented with fitting local models, planning with them to generate actions, and using these actions as supervision to fit a policy, then iterating to get better and better models around the areas that the policy needs. This has been successfully applied in *end-to-end learning* [61].

Another way to reduce the number of real-world samples required for learning is to reuse information from previous learning episodes on other tasks, rather than starting from scratch, known as *transfer learning*. In RL, agents rely heavily on prior experience when learning a new task and a common approach in RL is to learn an "easy" skill before fitting the model to learn something far

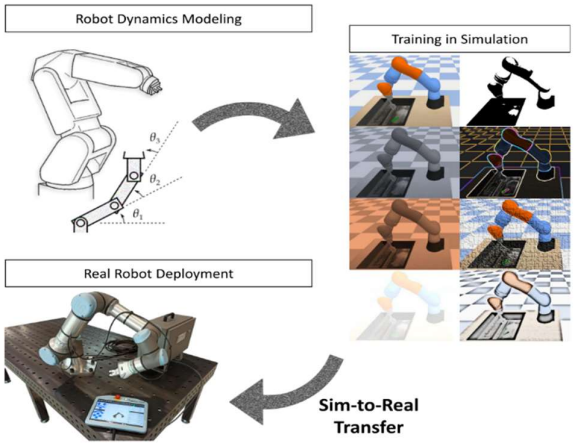

Fig. 5. Conceptual view of a sim-to-real transfer process as presented in [34]. One of the most common methods when training in simulation is domain randomization.

more complicated. Pertsch et al. addressed the problem of agents that possess a full set of available skills (during learning, not all skills should be explored with equal probability). They proposed a deep latent variable model, SPiRL (Skill-Prior RL), that jointly learns an embedding space of skills [62].

3.5. State-of-the-Art DRL Algorithms in Robotics

Policy gradient-based actor-critic algorithms are amongst the most popular algorithms in the reinforcement learning framework due to their advantage of being able to search for optimal policies using low-variance gradient estimates.

The variety of DRL architectures is partitioned into two different branches: discrete action space algorithms (DAS), such as Deep Q-Network (DQN), and continuous action space algorithms (CAS), such as Deep Deterministic Policy Gradient (DDPG). Further, the CAS algorithms are divided into stochastic continuous action space (SCAS) and deterministic continuous action space (DCAS) algorithms. Figure 6 illustrates a hierarchical view of DRL algorithms.

An actor-critic DRL agent with experience replay that is stable, sample efficient, and performs remarkably well in challenging environments and continuous control problems was presented in [63]. Replay is a valuable tool for improving sample efficiency. Experience replay has gained popularity in deep Q-learning where it is often used for reducing sample correlation.

Another actor-critic method was proposed in [64], called ACKTR. It applies trust region optimization to DRL using Kronecker-factored approximation to the curvature. Work [65] presented an actor-critic, model-free algorithm based on deterministic policy gradient that can operate over continuous action spaces, called DDPG.

DDPGwB was introduced in [66], as a novel algorithm that utilizes one or more base controllers to efficiently and safely learn challenging sparse-reward robotic tasks, with the learned state-based or image-based policies exceeding the performances of the base controllers. The DDPGwB algorithm is built upon DDPG and incorporates the controllers into stages of exploration, Q-value estimation as well as policy update.

Hindsight Experience Replay (HER) [67] allows sample-efficient learning from rewards which are sparse and binary, avoiding the need for complicated reward engineering. New methods for RL, called PPO were presented in [68], offering the possibility to alternate between sampling data through interaction with the environment, and optimizing a surrogate objective function using stochastic gradient ascent. Notable results can be achieved by combining model-free and model-based DRL with Guided Policy Search (GPS) [69].

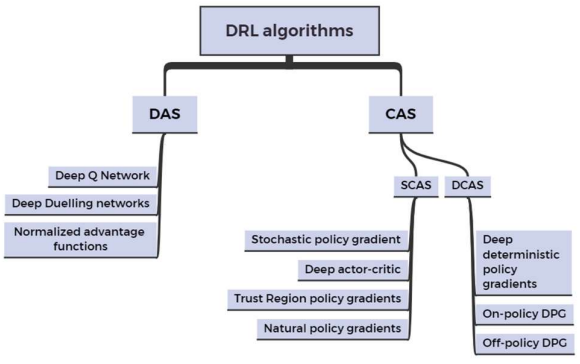

Fig. 6. The DRL topology.

When the RL agent explores the environment randomly, it results in low exploration efficiency, especially in robotic manipulation tasks with high dimensional continuous state and action space. In [70] the Augmented Curiosity-Driven Experience Replay (ACDER) method was proposed, which leverages a new goal-oriented curiosity-driven exploration to encourage the agent to pursue novel and task-relevant states more purposefully and the dynamic initial states selection as an automatic exploratory curriculum to further improve the sample-efficiency. Their approach complements HER by introducing a new way to pursue valuable states. The policies trained in simulation for reach, push and pick&place tasks perform well on the physical robot without any additional fine-tuning.

The problem of inverse RL is relevant to a variety of tasks including value alignment and robot LfD. [71] presented an inverse RL framework called Bayesian optimization-IRL (BO-IRL) which identifies multiple solutions that are consistent with the expert demonstrations by efficiently exploring the reward function space. Empirical results on synthetic and real-world environments (model-free and model-based) show that BO-IRL discovers multiple reward functions while minimizing the number of expensive exact policy optimizations.

Ho and Ermon [72], proposed a novel framework, called GAIL, for directly extracting a policy from data, in an inverse RL fashion. Another state-of-the-art policy-search RL algorithm, called PI2 (policy improvement with path integrals), was proposed by Theodorou et al. in [73], for learning parameterized control policies based on the framework of stochastic optimal control with path integrals. The approach demonstrates significant performance improvements over gradient-based policy learning and scalability to high-dimensional control problems.

A model-based RL framework called Critic PI2, which combines the benefits from trajectory optimization, deep actor-critic learning, and model-based reinforcement learning was presented in [74]. Empirical experiments demonstrated that Critic PI2 achieved a new state of the art in a range of challenging continuous domains, and that planning with a critic significantly increases the sample efficiency and real-time performance.

Focusing on the robotics field needs, [75] proposed the use of RL to learn how to compose hierarchical object-centric controllers for manipulation tasks. Their approach has several advantages: the object-centric controllers can be reused across multiple tasks, and controller compositions are invariant to certain object properties. Experiments showed that the proposed approach leads to more guided exploration and consequently improved sample efficiency, and it enables zero-shot generalization to test environments and simulation-to-reality transfer without fine-tuning.

Another approach, called COG, [76] showed that prior data can be reused to extend new skills simply through dynamic programming. Addressing the skill transfer problem, a novel approach, called SPiRL (Skill-Prior RL) was recently proposed in [62]. A new contextual off-policy RL algorithm, named LAtent-Movements Policy Optimization (LAMPO) was recently introduced in [77]. LAMPO can provide gradient estimates from previous experience using self-normalized importance sampling, hence, making full use of samples collected in previous learning iterations.

To address the issues RL algorithms face when task rewards are sparse, a novel form of intrinsic motivation that can allow robotic manipulators to learn useful manipulation skills with only sparse extrinsic rewards was proposed in [78]. Through maximizing the mutual dependence between robot actions and environment states, namely *the empowerment*, this intrinsic motivation helps the agent to focus more on the states where it can effectively "control" the environment instead of the parts where its actions cause random and unpredictable consequences. Empirical evaluations in different robotic manipulation environments with different shapes of the target object demonstrate the advantages of this empowerment-based intrinsic motivation over other state-of-the-art solutions to sparse-reward RL tasks.

Developments in the robotic vision field were also enabled by state-of-the-art methods, such as presented in [79], where DRL base system for controlling a robotic manipulator with only visual perception was presented. Levine et al. presented a learning-based approach to hand-eye coordination for robotic grasping in [60].

4. ROBOT VISION

In a simplified view, computer vision takes images and translates them into information, while robotic vision translates images into actions. Some of the main challenges for robotic vision are active learning, uncertainty estimation, and active manipulation on spatial embodiment. To get the full grasp of limits and potentials of robotic vision, refer to [80].

Vision-based RL and imitation learning methods incorporating deep neural network structure can express complex behaviors, and they solve robotics manipulation tasks in an end-to-end fashion.

Miyajima presented the efficiency of DL when it is used for object detection, object pose estimation and robot grasp generation in [81]. The peg-hole insertion problem was addressed in [82] where DRL was used to learn the policy (which only takes RGB-D and joint information) end-to-end and then transfer the learned model to the real robot. They demonstrated that a purely eye-in-hand, image-based controller can be trained in simulation to perform peg-hole insertion with sub-centimeter accuracy.

By focusing on the task of object localization problem in [83] it was demonstrated that an object detector trained only in simulation can achieve high enough accuracy in the real world to perform grasping in clutter. *Image augmentation* is crucial for successful transfer from simulation to the real-world.

A novel approach to crossing the visual reality gap that uses no real-world data called Randomized to- Canonical

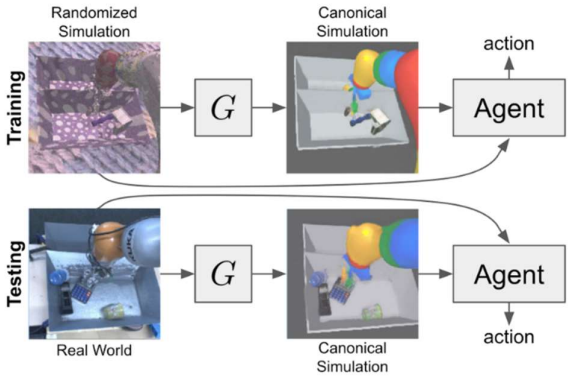

Fig. 7. Randomized simulation images are translated by a generator to a canonical simulation version and then used to train a robot grasping agent (top). The Sim-to-Real transfer of the agent (bottom) is presented in [84].

Adaptation Networks (RCANs) was presented in [84], and illustrated in Fig. 7. The authors demonstrated the effectiveness of the sim-to-real approach by training a vision-based closed-loop grasping RL agent in simulation, and then transferring it to the real world to attain 70% zero-shot grasp success on unseen objects, a result that almost doubles the success of learning the same task directly on *domain randomization* alone.

An approach to RL was proposed in [85] without hand-programmed reward functions by enabling a robot to learn from a modest number of examples of successful outcomes, followed by actively solicited queries (for only a tiny fraction of the states), where the robot shows the user a state and asks for a label to determine whether that state represents successful completion of the task. Their experiments show that the proposed method (VICE-RAQ) effectively learns to arrange objects, place books, and drape cloth, directly from images and without any manually specified reward functions, and with only 1–4 hours of interaction with the real world.

An approach capable of extracting dense reward functions algorithmically from robots' high-dimensional observations, such as images and tactile feedback was presented in [86]. Their approach learns rewards by estimating task progress in a self-supervised manner. They demonstrated the effectiveness and efficiency of their approach on two contact-rich manipulation tasks, namely, peg-in-hole and USB insertion. The experimental results indicate that the policies trained with the learned reward function achieves better performance and faster convergence compared to the baselines.

In [87], the authors presented ROLL (Reinforcement learning with Object Level Learning), a goal-conditioned visual self-supervised RL algorithm that incorporates object reasoning. The algorithm uses unknown object segmentation to ignore distractors in the scene for better reward computation and goal generation; occlusion reasoning is further enabled by employing a novel auxiliary loss and training scheme.

A method for offline learning of counterfactual predictions was proposed in [88]. Their approach combines offline and online learning in the use of counterfactual predictions learned offline to accelerate

online policy learning. Experiments were conducted in both simulation and real-world scenarios for evaluation.

RetinaGAN was introduced in [89], as an object-aware sim-to-real adaptation technique which transfers robustly across environments and tasks, even with limited real data. Retina-GAN involves a CycleGAN that adapts simulated images to look more realistic while also resulting in consistent objects predictions. This method provides reliable sim-to-real transfer for tasks in diverse visual environments. Addressing the vision sim-to-real gap, [90] introduced RL-CycleGAN. They proposed a method to employ generative models to translate simulated images into realistic ones and introduced a RL-scene consistency loss for image translation, which ensures that the translation operation is more robust.

5. SIMULATION FRAMEWORKS AND ENVIRONMENTS

As mentioned in the Section II, a first approach to solve diverse robot arm manipulation tasks can be done mainly in simulation, but it can happen often for the model to under-perform in the real world. Another approach is to train directly on the real robot, but this is difficult and has several drawbacks. A third strategy, which is the most promising, is to combine the two approaches, by pre-training models in simulation and continue learning in the real world.

OpenAI Gym is probably the most popular environment for benchmarking DRL algorithms. It includes a suite of robotics environments based on the MuJoCo simulation engine. Multi-Goal RL tasks can be developed for pushing, sliding and pick & place with a Fetch robotic arm, as well as in-hand object manipulation with a Shadow Dexterous Hand [4].

Acme [91] is a framework for distributed RL introduced by DeepMind. The framework is used to build readable, efficient, research oriented RL algorithms. At its core, Acme is designed to enable simple descriptions of RL agents that can be run at various scales of execution, including distributed agents.

Surreal [92], is an open-source, scalable framework that supports state-of-the-art distributed RL algorithms, known also as Scalable Robotic RL Algorithms. The framework decomposes a distributed RL algorithm into four components, which are a generation of experience (actors), storage of experience (buffer), updating parameters from experience (learner), and storage of parameters. It is a principled distributed learning formulation that accommodates both on-policy and off-policy learning.

Recently, Lucchi et al. introduced *robo-gym*, an open source and freely available framework (illustrated in Fig. 8.) that allows to train DRL control policies in distributed simulations and later, to apply them directly on the real world robots [93]. As the toolkit will continuously grow, it will serve as a solid base for developing research within the field of DRL in robotics.

Hein et al. designed a benchmark that bridges the gap between freely available, documented, and motivated artificial benchmarks and properties of real industrial problems. The Industrial Benchmark (IB) [94] is a useful addition to the set of existing RL benchmarks, addressing

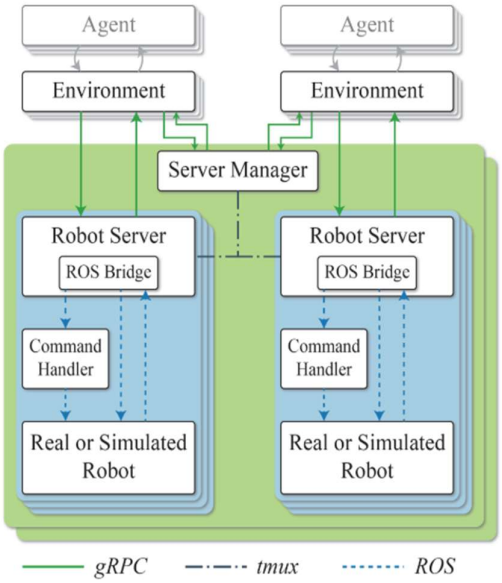

Fig. 8. The *robo-gym* framework as presented in [93].

a particular set of properties, such as stochastic dynamic with high dimensional continuous state and action spaces, motivated by industrial control problems.

6. CONCLUSIONS

This work presents a comprehensive review of RL and DRL approaches focusing on their applications in robotic assembly. The main challenges RL faces in robotics such as the necessity of large training data, long training times, model uncertainty, safety concerns and reality gaps are discussed and solutions to overcome them are offered.

For robotic manipulation, continuous action domain algorithms are the most effective and applicable. The capacity to self-optimize the controllers of robot arms is crucial in the Industry 4.0 setting. This capability is mandatory for handling the frequent changes that occur in the manufacturing process, to ensure high accuracy and precision, and therefore to guarantee cost efficiency and high quality of the manufactured products. For applications that require high positioning accuracy, fine-tuning the controller for each task will be unfeasible, thus a self-learning capability will be crucial.

In the context of smart factories, for a successfully assembled product, a RL-based assembly approach should plan the correct sequences of assembly tasks, the individual collision-free movements, compute the required forces and torques, and estimate the pose of assembly parts. Key strategies such as hierarchical task decompositions and skill reusability could ensure the successful use of RL-based methods in an industrial setting.

Despite significant advancements of RL in simulated domains such as games, its potentially great influence on real robot applications is still limited. There is a need to learn highly complicated reward functions and methods to represent highly skilled behaviors and skills. This opens the possibility for a trend towards exploration of sample efficient and time efficient algorithms, solving both continuous state and action space problems. Addressing

the need for large, difficult to collect and expensive training data for learning, methods and approaches have emerged to facilitate the learning process for robots, such as data augmentation, domain randomization, and sim-to-real transfer.

This work includes a brief introduction to RL simulation frameworks and environments that are currently freely available and could enable further research opportunities.

DL techniques have also changed many aspects of computer vision over the past decade and have been rapidly adopted into robotics as well. However, robotic perception, robotic learning, and robotic control are demanding tasks that continue to pose serious challenges on the techniques typically used. Nevertheless, the use of robotic vision in various robotic assembly tasks presents potential development directions leading to better performance and faster convergence compared to current the baselines.

Answering the question whether or to what extent, can the knowledge of solving one task help in solving another, may help in achieving automatic transfer learning, including the source task selection, the mapping function design, and more. This will accelerate the learning process and could become an essential topic in RL.

ACKNOWLEDGEMENTS: This work has been funded by the European Social Fund from the Sectorial Operational Programme Human Capital 2014-2020, through the Financial Agreement with the title "Scholarships for entrepreneurial education among doctoral students and postdoctoral researchers (Be Entrepreneur!)", Contract no. 51680/09.07.2019 - SMIS code: 124539.

REFERENCES

[1] J. Xiao, L. Li, T. Zhang, and Y. Zou, "Time-Optimal Path Tracking for Industrial Robots: A Dynamic Model-Free Reinforcement Learning Approach," *arXiv*, Jul. 2019.

[2] F. Rubio, F. Valero, J. Sunyer, and J. Cuadrado, "Optimal time trajectories for industrial robots with torque, power, jerk and energy consumed constraints," *Ind. Rob.*, vol. 39, no. 1, pp. 92–100, 2012.

[3] J. Yamada et al., "Motion Planner Augmented Reinforcement Learning for Robot Manipulation in Obstructed Environments," Oct. 2020.

[4] M. Plappert et al., "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research," *arXiv*, Feb. 2018.

[5] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," 2020.

[6] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Rob. Auton. Syst.*, vol. 112, pp. 72–83, Feb. 2019.

[7] D. Schwung, F. Csaplar, A. Schwung, and S. X. Ding, "An application of reinforcement learning algorithms to industrial multi-robot stations for cooperative handling operation," 2017.

[8] D. Kragic and M. Vincze, "Vision for Robotics," *Found. Trends Robot.*, vol. 1, no. 1, pp. 1–78, 2009.

[9] A. K. Bozkurt, Y. Wang, and M. Pajic, "Secure Planning Against Stealthy Attacks via Model-Free Reinforcement Learning," Nov. 2020.

[10] S. Yin, W. Ji, and L. Wang, "A machine learning based energy efficient trajectory planning approach for industrial robots," *Procedia CIRP*, vol. 81, pp. 429–434, 2019.

[11] A. Rupam Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, "Setting up a Reinforcement Learning Task with a Real-World Robot," in *IEEE International Conference on Intelligent Robots and Systems*, Mar. 2018, pp. 4635–4640.

[12] G. B. Murali, B. Deepak, B. Biswal, G. B. Mohanta, and A. Rout, "Robotic Optimal Assembly Sequence Using Improved Cuckoo Search Algorithm," *Procedia Comput. Sci.*, vol. 133, pp. 323–330, 2018.

[13] Y. Su, H. Mao, and X. Tang, "Algorithms for solving assembly sequence planning problems," *Neural Comput. Appl.*, pp. 1–10, Jun. 2020.

[14] G. B. Murali, B. B. V. L. Deepak, and B. B. Biswal, "Optimal robotic assembly sequence planning using crab shell search algorithm," *Int. J. Mechatronics Autom.*, vol. 7, no. 3, p. 147, 2020.

[15] G. B. Murali, B. B. V. L. Deepak, B. B. Biswal, and Y. Karun Kumar, "Robotic Assembly Sequence Generation Using Improved Fruit Fly Algorithm," in *Lecture Notes in Mechanical Engineering*, 2020, pp. 239–247.

[16] M. V. A. Raju Bahubalendruni, B. B. Biswal, M. Kumar, and R. Nayak, "Influence of assembly predicate consideration on optimal assembly sequence generation," *Assem. Autom.*, vol. 35, no. 4, pp. 309–316, Sep. 2015.

[17] K. Watanabe and S. Inada, "Search algorithm of the assembly sequence of products by using past learning results," *Int. J. Prod. Econ.*, vol. 226, p. 107615, Aug. 2020.

[18] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intell. Serv. Robot.*, vol. 6, no. 1, pp. 33–51, Jan. 2013.

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

[20] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annu. Rev. Control. Robot. Auton. Syst.*, vol. 3, no. 1, pp. 297–330, May 2020.

[21] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, Jan. 2003.

[22] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding Neural Networks Through Deep Visualization," Jun. 2015.

[23] Chun-Gui Li, Meng Wang, and Qing-Neng Yuan, "A Multi-agent Reinforcement Learning using Actor-Critic methods," in *2008 International Conference on Machine Learning and Cybernetics*, Jul. 2008, vol. 2, pp. 878–882.

[24] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control," *arXiv*, Dec. 2018.

[25] X. Lyu, S. Li, S. Siriya, Y. Pu, and M. Chen, "MBB: Model-Based Value Initialization for Reinforcement Learning," 2020.

[26] Y. H. Wang, T. H. S. Li, and C. J. Lin, "Backward Q-learning: The combination of Sarsa algorithm and Q-learning," *Eng. Appl. Artif. Intell.*, vol. 26, no. 9, pp. 2184–2193, Oct. 2013.

[27] M. Coggan, "Exploration and exploitation in reinforcement learning," *Res. supervised by Prof. Doina Precup, CRA-W*, 2004.

[28] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," 2003.

[29] 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," *Int. J. Rob. Res.*, vol. 37, no. 4–5, pp. 421–436, Apr. 2018.

[30] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," 2018.

[31] T. Dai, H. Liu, and A. Anthony Bharath, "Episodic Self-Imitation Learning with Hindsight," *Electronics*, vol. 9, no. 10, p. 1742, Nov. 2020.

[32] I. Exarchos, Y. Jiang, W. Yu, and C. K. Liu, "Policy Transfer via Kinematic Domain Randomization and Adaptation," Nov. 2020.

[33] M. S. A. Hameed, M. M. Khan, and A. Schwung, "Curiosity Based Reinforcement Learning on Robot Manufacturing Cell," Nov. 2020.

[34] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey," Sep. 2020.

[35] N. Hansen and X. Wang, "Generalization in Reinforcement Learning by Soft Data Augmentation," Nov. 2020.

[36] F. Yang, C. Yang, D. Guo, H. Liu, and F. Sun, "Fault-Aware Robust Control via Adversarial Reinforcement Learning," Nov. 2020.

[37] Y. S. Shao, C. Chao, S. Kousik, and R. Vasudevan, "Reachability-based Trajectory Safeguard (RTS): A Safe and Fast Reinforcement Learning Safety Layer for Continuous Control," Nov. 2020.

[38] V. Kumar, D. Hoeller, B. Sundaralingam, J. Tremblay, and S. Birchfield, "Joint Space Control via Deep Reinforcement Learning," Nov. 2020.

[39] J. C. Kiemel and T. Kröger, "Learning Robot Trajectories subject to Kinematic Joint Constraints," Nov. 2020.

[40] J. Kurosu, A. Yorozu, and M. Takahashi, "Simultaneous dual-arm motion planning for minimizing operation time," *Appl. Sci.*, vol. 7, no. 12, p. 1210, Nov. 2017.

[41] S. S. Mirrazavi Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *Int. J. Rob. Res.*, vol. 37, no. 10, pp. 1205–1232, Sep. 2018.

[42] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2005, pp. 430–435.

[43] H. Ha, J. Xu, and S. Song, "Learning a Decentralized Multi-arm Motion Planner," Nov. 2020.

[44] D. Liberzon, *Calculus of Variations and Optimal Control Theory*. Princeton University Press, 2019.

[45] W. Li and E. Todorov, "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems," in *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, 2011, pp. 222–229.

[46] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the Sample Complexity of the Linear Quadratic Regulator," *Found. Comput. Math.*, vol. 20, no. 4, pp. 633–679, Aug. 2020.

[47] P. Kapoor, A. Balakrishnan, and J. V. Deshmukh, "Model-based Reinforcement Learning from Signal Temporal Logic Specifications," Nov. 2020.

[48] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Learning Stable Normalizing-Flow Control for Robotic Manipulation," Oct. 2020.

[49] A. Perrusquía, W. Yu, and A. Soria, "Position/force control of robot manipulators using reinforcement learning," *Ind. Rob.*, vol. 46, no. 2, pp. 267–280, Mar. 2019.

[50] N. Hogan, "Impedance control: An approach to manipulation: Part II-implementation," *J. Dyn. Syst. Meas. Control. Trans. ASME*, 1985.

[51] J. Luo et al., "Reinforcement learning on variable impedance controller for high-precision robotic assembly," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2019-May, pp. 3080–3087, Mar. 2019.

[52] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE J. Robot. Autom.*, 1987.

[53] W. D. Fisher and M. S. Mujtaba, "Hybrid position/force control. A correct formulation," *Int. J. Rob. Res.*, 1992.

[54] W. Gueaieb, F. Karray, and S. Al-Sharhan, "A robust hybrid intelligent position/force control scheme for cooperative manipulators," *IEEE/ASME Trans. Mechatronics*, 2007.

[55] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual Model Control : An Intuitive Approach," *Int. J. Rob. Res.*, 2001.

[56] L. Roveda, G. Pallucca, N. Pedrocchi, F. Braghin, and L. M. Tosatti, "Iterative Learning Procedure with Reinforcement for High-Accuracy Force Tracking in Robotized Tasks," *IEEE Trans. Ind. Informatics*, vol. 14, no. 4, pp. 1753–1763, Apr. 2018.

[57] M. Focchi et al., "Robot impedance control and passivity analysis with inner torque and velocity feedback loops," *Control Theory Technol.*, vol. 14, no. 2, pp. 97–112, 2016.

[58] L. Wang, Y. Xiang, and D. Fox, "Goal-Auxiliary Actor-Critic for 6D Robotic Grasping with Point Clouds," *Appl. Sci.*, vol. 10, no. 19, p. 6923, Oct. 2020.

[59] C. C. Beltran-Hernandez et al., "Learning Force Control for Contact-rich Manipulation Tasks with Rigid Position-controlled Robots," Mar. 2020.

[60] 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," *Int. J. Rob. Res.*, vol. 37, no. 4–5, pp. 421–436, Apr. 2018.

[61] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-End Robotic Reinforcement Learning without Reward Engineering," 2019.

[62] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating Reinforcement Learning with Learned Skill Priors," Oct. 2020.

[63] Z. Wang et al., "Sample efficient actor-critic with experience replay," *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, Nov. 2017.

[64] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation," 2017.

[65] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, Sep. 2016.

[66] M. Xin, G. Wang, Z. Liu, and H. Wang, "Achieving Sample-Efficient and Online-Training-Safe Deep Reinforcement Learning with Base Controllers," 2020.

[67] M. Andrychowicz et al., "Hindsight experience replay," 2017.

[68] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[69] A. Franceschetti, E. Tosello, N. Castaman, and S. Ghidoni, "Robotic Arm Control and Task Training through Deep Reinforcement Learning," *arXiv*, May 2020.

[70] B. Li, T. Lu, J. Li, N. Lu, Y. Cai, and S. Wang, "ACDER: Augmented Curiosity-Driven Experience Replay," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 4218–4224, Nov. 2020.

[71] S. Balakrishnan, Q. P. Nguyen, B. K. H. Low, and H. Soh, "Efficient Exploration of Reward Functions in Inverse

Reinforcement Learning via Bayesian Optimization," Nov. 2020.

[72] J. Ho and S. Ermon, "Generative adversarial imitation learning," 2016.

[73] E. Theodorou, J. Buchli, and S. Schaal, "Learning policy improvements with path integrals," 2010.

[74] J. Fan, H. Ba, X. Guo, and J. Hao, "Critic PI2: Master Continuous Planning via Policy Improvement with Path Integrals and Deep Actor-Critic Reinforcement Learning," Nov. 2020.

[75] M. Sharma, J. Liang, J. Zhao, A. LaGrassa, and O. Kroemer, "Learning to Compose Hierarchical Object-Centric Controllers for Robotic Manipulation," Nov. 2020.

[76] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine, "COG: Connecting New Skills to Past Experience with Offline Reinforcement Learning," Oct. 2020.

[77] S. Tosatto, G. Chalvatzaki, and J. Peters, "Contextual Latent-Movements Off-Policy Optimization for Robotic Manipulation Skills," Oct. 2020.

[78] S. Dai, W. Xu, A. Hofmann, and B. Williams, "An Empowerment-based Solution to Robotic Manipulation Tasks with Sparse Rewards," Oct. 2020.

[79] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *Australas. Conf. Robot. Autom. ACRA*, Nov. 2015.

[80] N. Sünderhauf et al., "The limits and potentials of deep learning for robotics," *Int. J. Rob. Res.*, vol. 37, no. 4–5, pp. 405–420, Apr. 2018.

[81] R. Miyajima, "Deep learning triggers a new era in industrial robotics," *IEEE Multimed.*, vol. 24, no. 4, pp. 91–96, Oct. 2017.

[82] F. Chervinskii, A. Rybnikov, D. Bogunowicz, and K. Vendidandi, "Sim2Real for peg-hole insertion with eye-in-hand camera," *arXiv*, May 2020.

[83] 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," Mar. 2017.

[84] S. James et al., "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 12619–12629, Dec. 2019.

[85] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-End robotic reinforcement learning without reward engineering," *arXiv*. 2019, doi: 10.15607/rss.2019.xv.073.

[86] Z. Wu, W. Lian, V. Unhelkar, M. Tomizuka, and S. Schaal, "Learning Dense Rewards for Contact-Rich Manipulation Tasks," Nov. 2020.

[87] Y. Wang, G. N. Narasimhan, X. Lin, B. Okorn, and D. Held, "ROLL: Visual Self-Supervised Reinforcement Learning with Object Reasoning," Nov. 2020.

[88] J. Jin, D. Graves, C. Haigh, J. Luo, and M. Jagersand, "Offline Learning of Counterfactual Perception as Prediction for Real-World Robotic Reinforcement Learning," Nov. 2020.

[89] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer," Nov. 2020.

[90] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "RL-CycleGan: Reinforcement learning aware simulation-to-real," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 11154–11163, Jun. 2020.

[91] M. Hoffman et al., "Acme: A Research Framework for Distributed Reinforcement Learning," 2020.

[92] L. Fan* et al., "SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark," 2018.

[93] M. Lucchi, F. Zindler, S. Mühlbacher-Karrer, and H. Pichler, "Robo-gym -An Open Source Toolkit for Distributed Deep Reinforcement Learning on Real and Simulated Robots," *arXiv*, Jul. 2020.

[94] D. Hein et al., "A benchmark environment motivated by industrial control problems," *2017 IEEE Symp. Ser. Comput. Intell. SSCI 2017 - Proc.*, vol. 2018-Janua, pp. 1–8, 2018.