robotics

MDPI

Review

Reinforcement Learning for Pick and Place Operations in Robotics: A Survey

Andrew Lobbezoo *, Yanjun Qian and Hyock-Ju Kwon

AI for Manufacturing Laboratory, Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada; y32qian@uwaterloo.ca (Y.Q.); hjkwon@uwaterloo.ca (H.-J.K.)
* Correspondence: ajlobbez@uwaterloo.ca

Abstract: The field of robotics has been rapidly developing in recent years, and the work related to training robotic agents with reinforcement learning has been a major focus of research. This survey reviews the application of reinforcement learning for pick-and-place operations, a task that a logistics robot can be trained to complete without support from a robotics engineer. To introduce this topic, we first review the fundamentals of reinforcement learning and various methods of policy optimization, such as value iteration and policy search. Next, factors which have an impact on the pick-and-place task, such as reward shaping, imitation learning, pose estimation, and simulation environment are examined. Following the review of the fundamentals and key factors for reinforcement learning, we present an extensive review of all methods implemented by researchers in the field to date. The strengths and weaknesses of each method from literature are discussed, and details about the contribution of each manuscript to the field are reviewed. The concluding critical discussion of the available literature, and the summary of open problems indicates that experiment validation, model generalization, and grasp pose selection are topics that require additional research.

Keywords: reinforcement learning; Markov decision process; policy optimization; robotic control; simulation environment; pose estimation; imitation learning

Citation: Lobbezoo, A.; Qian, Y.; Kwon, H.-J. Reinforcement Learning for Pick and Place Operations in Robotics: A Survey. *Robotics* **2021**, *10*, 105. https://doi.org/10.3390/robotics10030105

Academic Editors: Giuseppe Carbone and Alessandro Di Nuovo

Received: 28 July 2021
Accepted: 6 September 2021
Published: 13 September 2021

1. Introduction

Robotic arms have been used in industry for years to automate repetitive, strenuous, and complex tasks in which speed and precision are critical. The difficulty with the conventional control of robotic arms is that they require individual programming on a task-by-task basis with no margin for error. As a result, task programming requires extensive effort by a robotics engineer or tedious online manipulation by skilled robot operators with the control pendant [1–3].

1.1. Robotics Background

With traditional robotic control strategies, motion planning is founded on forward and inverse kinematics. The forward kinematics combine the direct measurement of all joint orientations with the lengths of linkages connected to the joints to determine the end-effector position. Similarly, inverse kinematics establish the values of joint positions required to place the end effector at a new location with a new orientation. The Jacobian of the forward kinematics and the inverse Jacobian for the inverse kinematics are used to determine the velocity of the end effector at different points over the motion path [4,5].

Typically, an engineer does not directly develop the code to apply the forward and inverse kinematics to complete each robotic task. Instead, graphical or text-based programming environments implement interpreters and compilers to translate basic commands or position controls to the robot in its native language [3,6]. The programming environment is determined by the interfaces provided by the supplier and the choice between online and offline programming [7].

Online programming is traditionally used in industry to allow skilled operators to teach basic motion trajectories and operations to the robot using teach-pendant control interfaces or manual robotic arm positioning. There are many difficulties associated with online programming, such as robot downtime, range in pendant interfaces, and lack of dynamic control to handle changes in the environment. Online programming is only usable for basic tasks with high repeatability. Reprogramming is required for any minor variances in the workspace, limiting the applicability for this type of programming [7,8].

Offline programming is the alternative control technique. This approach involves skilled programmers interacting with graphical program environments and/or text-based programming languages to develop control strategies for the robotic arm. Offline programming has the advantages of reducing downtime (and the associated downtime cost) and allowing the implementation of advanced control strategies [3,7,8]. The disadvantages of offline programming include the requirement for programmers to have experience with generic or controller specific languages, and/or the graphical model based control environment selected by the manufacturer [3,7].

Due to the difficulties with both the online and offline approaches, robotic implementation in small and medium-sized labor-dependent enterprises in industry has been slow. The earliest forms of electric robotic arms driven with forward and inverse kinematics were developed in the early 1970s [9]. Nearly 50 years since this development, only 33.96% of small and medium firms have implemented any form of robotics, a value which is 35.64% less than large firms (statistic taken in 2015) [10]. Automation improves productivity and reduces labor costs [10], which puts small and medium-sized enterprises at a significant disadvantage. The barrier for small companies to incorporate robotics is the high cost of implementing and reprogramming robots and the inability of robots to handle environments that change dynamically [11].

To reduce difficulty with the implementation of robotics in factory settings, reinforcement learning (RL) is becoming a popular alternative to task-specific programming. The goal of RL in robotics is to enable the agent (the robotic arm) to complete basic operations without direct command programming, and to handle changes in the task without requiring reprogramming. The logistics action of pick-and-place is an obvious example of a task which a robotic agent should be able to perform with RL.

To complete pick-and-place, a robotic arm must select a target object from a cluttered or isolated location and place the object at a specified position with a particular orientation (Figure 1) This generic task is applicable for various sorting applications which traditionally require human labor. Of the papers published on the topic of pick-and-place with RL (Section 8), implementations differ drastically. To date, no single approach has been selected as the industry standard.

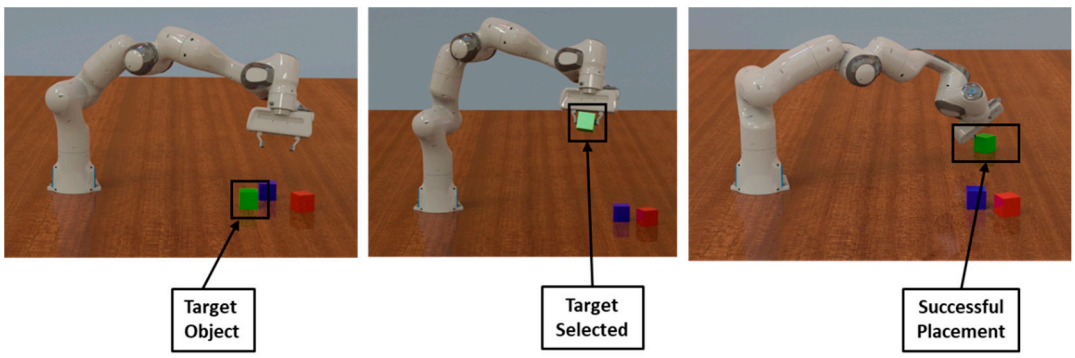

Figure 1. Pick-and-place of color-coded square blocks with the Franka Emika Panda robot.

1.2. Related Work

A broad-scale generalized strategy for implementing pick-and-place with RL in robotics is needed before this technique can be implemented at scale. Mohammed and

Chua [12], Liu et al. [13], and Tai et al. [14] have all written review papers focused on RL in robotics; however, these papers have a broad range of focus in terms of robotic agents used and the task completed.

1.3. Scope of Review

This survey provides a holistic review of RL in robotics as it relates to the pick-and-place task. The fundamental formulation of RL is reviewed, and the elements unique to the robotics application are analyzed in detail. A comprehensive list of papers published on the topic are presented, and a critical analysis on the state of research is conducted. The section breakdown is as follows.

Section 2 reviews the fundamental framework for RL, as it can be found in the Markov decision process. After introducing the framework, this section examines the aspects of the robotics application that makes the problem difficult to solve. Section 3 discusses the applicable value function and policy search approaches to policy optimization. The mathematical formulation for each approach is presented and explained. Section 4 investigates the intuition behind reward shaping and highlight some critical techniques developed in the past 30 years. Section 5 explains inverse reinforcement learning and its value in determining complex underlying reward functions. Section 6 explores the intuition behind pose estimation, discussing both gripper and target object pose. Section 7 assesses the choices for simulation environment and reviews how this choice affects the end testing accuracy for robotic pick-and-place. Section 8 performs an extensive audit of the available publications related to RL pick-and-place, with the goal of providing direction for individuals working on this problem in the future. Section 9 includes a brief discussion on open problems.

2. RL Formulation

The goal for completing a pick-and-place operation without task-specific programming is to allow the agent to independently perform actions in the environment, then learn the optimal strategy for its future tasks based on its experiences [15]. The learning process is analogous to a child learning to walk. A child must first attempt the motions of moving limbs, rolling over, and crawling, before it can complete the target action of walking. The child's behavior is driven by rewards from its environment after the completion of each task. Positive affirmation from the child's parents is an example of an environmental reward which would drive the walking behavior.

2.1. Markov Decision Process

The Markov decision process (MDP) is the underlying formulation for learning with these reinforcement iterations [15]. The MDP encourages specific behaviors and discourages others by feeding rewards or punishments to the agent based on the agent's action in the environment. The agent selects an action based on its policy function for available actions $a \in \mathcal{A}(s)$ in each state $s \in \mathcal{S}$. In the deterministic case, the same action a is always taken in state s, and the policy is denoted as $\pi(s)$. For an agent operating in a stochastic environment, the policy defines the probabilities of selecting each possible action given a state. The policy for the stochastic case is denoted as $\pi(a|s)$ [15]. The primary goal for RL is to find the optimal policy $\pi^*(a|s)$. The optimal policy can be found through the implementation of the value function approach [16] or directly through a policy search (both discussed further in Section 3) [17].

The goal of the policy function is to optimize the expected return G, the (weighted) sum of all discounted returns before the final time step. At time step t, the expected return is expressed as [15].

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{1}$$

where T is the total time step, R_k is the reward for each step, and γ is the discounting parameter, a number between 0 and 1, which discounts future rewards. Figure 2 depicts the standard MDP for a robotic arm.

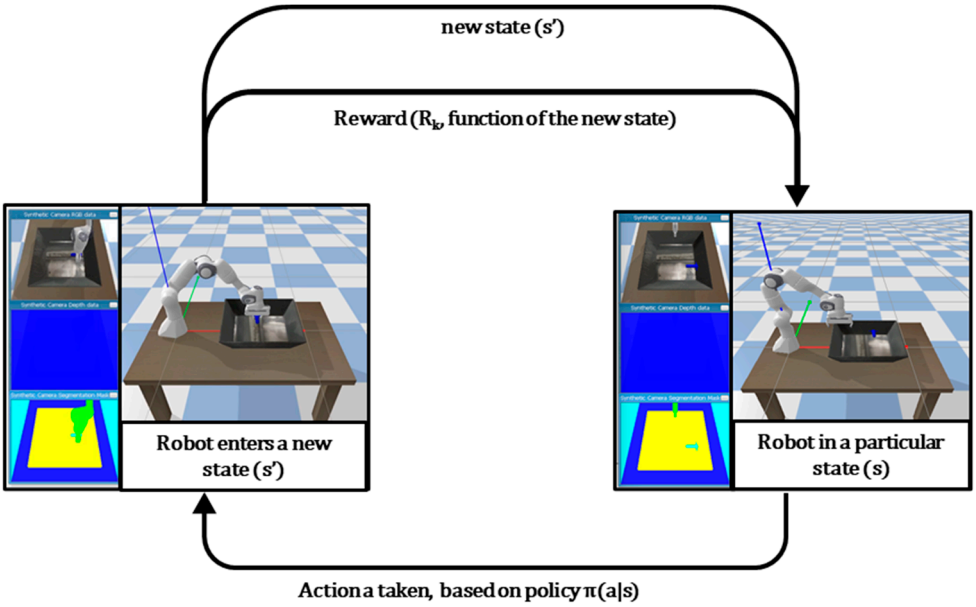

Figure 2. Agent-environment interaction in the MDP.

Again, the intuition behind the MDP can be understood with the analogy of a child learning to crawl. The policy function is analogous to the child's thought process before attempting an action. The child's policy is dependent on the reward it receives for completing the action and is the most significant factor which drives learning.

2.2. RL for Pick-and-Place in Robotics

RL formulated around the basic MDP appears to be intuitive and straightforward; however, robotic training has specific nuances. The primary impediments include the high dimensionality, continuous action space, and extensive training time.

Most robotic arms implement 6 degrees of freedom as this is the minimum freedom required to reach any position in space at any angle [18]. The freedom of movement makes the task complex, as all pick-and-place actions require the control of the torque and velocity of each joint of the agent. Additionally, the robot operates in a complex environment with a continuous operational space. Rather than having discrete state choices, the agent in this problem has infinitely many states to explore. The method of limiting the number of actions and states to a tractable set becomes a choice that affects the training time, test accuracy, and repeatability [19].

Many RL algorithms have been developed for the application with simple agents and environments. OpenAI (https://gym.openai.com/, accessed on 15 August 2021) has attempted to standardize the benchmarking environments for RL by creating open access simulations for testing control strategies. Cartpole is an example of a classic control problem found on OpenAI, in which the agent can make a command (left or right) and receive a low dimensional feedback signal (position and angular velocity) [20]. Compared to this simple benchmark, robotic arm manipulation is foundationally more difficult. The high dimensionality of robotic arms drives the need for complex reward functions to promote smooth motions and minimize training time [21]. Even with complex reward functions, the type of object picked, and the pose (orientation and position) of the target object affects the probability of task completion [22].

Due to the high dimensionality and continuous action space, the agent requires extensive training to learn the optimal policy. Online training is costly because of the required human supervision, robotic wear, danger of damaging the robot, and lack of available training robots. The costs associated with online training drives the need for simulation training [16]; however, simulation training often has a low test accuracy. The fundamental problem with simulation training is that the knowledge trained in the simulation must

be transferred to the real world. Any inaccuracies in modeling physical parameters such as shapes and weights of objects handled, lighting, or friction coefficients will cause test accuracies to be lower than training accuracies [23].

3. Policy Optimization

Finding the optimal policy for a RL problem involves value iteration or policy search. With both policy optimization techniques, the solution process changes depending on if the approach is model-based or model-free. Additionally, the exploration-exploitation tradeoff significantly influences the speed of solution convergence.

The model-based technique implements an understanding of the environment through prior learning or through state-space search to create an approximation of the transition probabilities between states given actions $p(s'|s,a)$) (also commonly formulated as $T(s'|a,\ s)$). The model-free (direct) design has no explicit representation of the system dynamics. Instead, the model-free technique learns the value function directly from the rewards received during training [15,24,25]. For RL in robotics, most approaches are model-free, since creating a perfect model of all transition probabilities in a continuous space is intractable.

Finding the policy for the agent always involves a tradeoff between searching random actions to find the optimal policy and greedily selecting the optimal action to improve rewards. In RL, ε is the variable used to represent the probability of not taking the reward-maximizing action during exploration [15]. To ensure that the whole action space is explored, the agent traditionally implements some form of on- or off-policy learning. On-policy learning improves the current policy marginally by selecting the optimal action for the majority $(1-\varepsilon)$ of actions and selecting a random action with a probability (ε) for the rest. Off-policy learning applies two policies; the target policy, which is the policy being learned about, and the behavior policy, which is the policy that drives behavior for learning [15].

In the following sections, the value function and policy search approaches are explained, and different search techniques inside each methodology are reviewed [25].

3.1. Value Function Approach

The value function approach is based on dynamic programming (DP). DP is a strategy of breaking up a complex problem into a sequence of successive sub-problems which can be iteratively solved. Implementing DP for RL requires an understanding of value functions.

3.1.1. Value Functions

Value functions are founded on the understanding that each state has an associated value dependent on the reward achieved in that state and that state's potential to be a step in achieving future rewards with the agent's current policy. A reward is often only received in a specific goal state, so positions proximal to the goal state have the highest value. The value function is found by breaking down the problem into successive sub-problems following the sequence of states. The Bellman equations in Equations (2) and (3) are established to solve for action-value function q_π, and state value function v_π respectively [15].

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \tag{2}$$

$$v_\pi(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \tag{3}$$

In stochastic reinforcement learning, randomness in the system implies that taking an action a from state s does not always result in the same final state s'. The transition dynamics (or transition probability) of the environment $p(s',r|s,a)$ is used to represent the probability of reaching state s' and receiving a reward r given that action a is taken in state s [15]. Using this formulation, the action-value function $q_\pi(s,a)$ given in Equation (2) can

be understood as being equal to the summation of the transition probabilities of reaching all possible states (s') multiplied by the value $(r + \gamma v_\pi(s'))$ of each possible state (s').

The state-value function given in Equation (3) can be understood similarly. The primary difference between the two formulations is that the state-value function (Equation (3)) incorporates action selection. The probability of selecting action a is dependent on the stochastic policy $\pi(a|s)$. Based on this understanding, the value of a given state is equal to the summation over all actions of the probability of selecting a particular action multiplied by the state value function $q_\pi(s, a)$.

In reinforcement learning, state- and action-value functions are implemented for finding the optimal policy with Bellman's optimality equations. The optimal state value function v_* and optimal action-value function q_* are found by maximizing the probability of choosing the actions which lead to states yielding values larger or equal to those from all other policies [15].

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s') \right] \tag{4}$$

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right] \tag{5}$$

For finite discrete actions and states, lookup tables can be used to pick appropriate actions for each state to form the optimal policy. For pick-and-place operations in robotics, determining Bellman's optimality equations is difficult because the space of all states is continuous. Typically the Monte Carlo (MC) or temporal difference (TD) approaches [16] are used. Both approaches are founded on dynamic programming (DP).

3.1.2. Dynamic Programming

DP is a model-based strategy that creates internal models for transition probabilities and rewards to calculate the value function. A primary assumption of DP is that the models perfectly represent reality, which is unrealistic for continuous spaces. Even though it cannot be directly applied for RL in robotics, DP is the framework for understanding the MC and TD techniques [15]. DP consists of two fundamental approaches: policy iteration and value iteration.

Policy iteration is a DP approach that consists of policy evaluation and policy improvement subtasks [26]. Policy evaluation is used to update the value of each state based on the current policy, transition probabilities, and the value of the following states [15].

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s') \right] \tag{6}$$

Policy evaluation is completed by calculating the $v_k(s)$ (Equation (6)) for all states with an arbitrarily initiated policy. After an infinite number of exploratory steps ($k \to \infty$), the transition probabilities between states given actions $p(s', r|s, a)$ and the state-values $v_k(s)$ can be determined. Policy improvement then greedily selects the best action that can be taken in each state based on these newly found transition probabilities and state values [15,16]. Policy improvement is solved by implementing $q_\pi(s, a)$ as shown [15]

$$\pi'(s) \doteq \operatorname*{argmax}_a q_\pi(s, a) \tag{7}$$

where argmax over actions involves taking the action which maximizes the value for each state. The new choice of action indicates the agent's policy in the next round of policy iteration. The process of policy iteration in DP consists of iterating between policy evaluation and policy improvement until the policy stops changing. At this point, the policy is assumed to be optimal [26].

Value iteration is an alternative to policy iteration, which can be used effectively by combining policy evaluation and improvement into one step. Value iteration is formulated as [15]

$$v_{k+1}(s) \doteq \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_k(s')\right] \tag{8}$$

with this formulation, the policy evaluation step is stopped after one iteration, and the policy for each state can be updated dynamically.

3.1.3. Model-Free Techniques

Techniques that are not model based can incorporate the DP strategy by using sample transitions and rewards to learn approximate value functions. Learning the value functions without complete probability distributions for all transitions requires exploration to ensure all states are visited, and non-greedy policies are investigated.

The Monte Carlo (MC) averaging returns approach is a model-free technique that requires no prior knowledge. The model only requires experience through interaction with the environment. MC completes various actions in each state, and the average returns for each action are used as the expected action reward. After many iterations, the solution will converge to the actual underlying value for each action [16].

MC approaches implement a combined on- and off-policy technique. The state values for MC are determined during off-policy environmental interaction where sample trajectories are explored. After the state values are determined, the policy is found during the on-policy policy improvement step of policy iteration. During this step, the greedy policy is found by taking the value-maximizing actions for the values learned during off-policy exploration [15].

The Temporal Difference (TD) approach is equivalent to an incremental implementation of MC. TD updates the estimates of the state-value function based on individual experiences without waiting for the outcome of the entire sample trajectory as opposed to MC. TD uses the difference between the old estimate of the value function and the experienced reward and new state to update the value function after one step, as shown in [15,25,26]:

$$V(s_t) \leftarrow V(s_t) + \alpha[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{9}$$

Here, α is the learning rate, a real number between 0 and 1 (usually 0.1 or smaller). This form of TD is referred to as the TD(0) algorithm.

State Action State Reward (SARSA) is a TD model which is used to update the action-value function ($Q(s, a)$) [16]. SARSA implements on-policy TD control, implying that the action value is updated after each exploration step [15,25].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * [R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{10}$$

Operating on-policy means that the action space is only explored due to the $\varepsilon - greedy$ aspect of on-policy search [15]. Due to its lack of off-policy exploration, SARSA does not have the same convergence guarantees as to the MC approach.

Q-learning is a method which is very popular in reinforcement learning research [15,25,26]. Q-learning is the off-policy equivalent of SARSA, formulated as: [15]

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{11}$$

The use of off-policy TD allows Q-learning to perform updates to the value function regardless of the action selected [25]. This choice guarantees the convergence of $Q(s_t, a_t)$ if the learning rate α is well defined [15].

Value function approaches are intuitive; however, they often struggle to converge due to the high dimensionality of robotic state space, the continuous nature of the problem, and the bootstrapped approaches used to estimate the value function quickly. Another

issue with the value function approach is that random exploration of the agent can result in mechanical damage due to range and torque limitations [27].

3.2. Policy Search Approach

Policy search (PS) is an approach to policy optimization which has recently shown potential in robotics as an alternative to the value function approach [16]. PS is a valuable tool in robotics due to the scalability with dimensionality, a vital issue with the value function approach [28]. Research has shown that PS is better than value function approaches for tasks that are finite horizon and are not repeated continuously [27].

PS does not approximate a policy as the value function does, i.e., by learning values of actions, then acting to optimize rewards based on these expected values. Instead, policy search implements a parameterized policy that can select the optimal action without reviewing the value function [15]. θ, the variable representing the policy parameters, could be understood intuitively as weights of the deep neural network that influences the policy for a value function [15,27] The formulation for the linear case is [27]

$$\pi_\theta(s) = \theta^T \, \Phi(s) \tag{12}$$

where $\Phi(x)$ is the basis function representing the state, and the policy parameters θ determine the choice of action in each state [27].

PS has the goal of learning the policy parameters that optimizes the performance measure $J(\theta)$, the expected cumulated discounted reward from a given state [27]. This reward is formulated as [15]

$$J(\theta) \doteq v_{\pi_\theta}(s) \tag{13}$$

wherein v_{π_θ} is the true value function for policy π_θ.

To maximize the cumulative reward, the optimal policy can be found using the policy gradient method [15], which implements gradient ascent by

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta) \tag{14}$$

where $\mu(s)$ is the on-policy probability distributions for policy π.

Like the value function approach, the policy search approach can be implemented with model-free and model-based techniques [16]. The model-free technique uses online agent actions to create trajectories from which to learn. This technique has the advantage of not needing to learn an accurate forward model, which is often more challenging to learn than the policy itself. The model-based technique is more sample conservative, as the agent creates internal simulations of the dynamics of the model based on a few observations from the online model. The agent then learns the policy based on these internal simulations [27]. The online model-free technique has been implemented more in research due to its ease of use; however, model-based approaches have shown better ability to generalize to unforeseen samples [27].

Inside of policy search, several methods have been developed and implemented in robotics, including REINFORCE, Actor-Critic, Deterministic Policy Gradient, and Proximal Policy Optimization.

3.2.1. REINFORCE

REINFORCE is a MC based policy gradient method that uses vector samples (episodes) filled with state, action, reward triples found with the policy π_θ to approximate G, the return as defined in Equation (1). The algorithm updates the policy parameters incrementally with [15]

$$\theta \leftarrow \theta + \alpha \, \gamma^t G \, \nabla \ln \pi_\theta(a_t|s_t, \, \theta) \tag{15}$$

where $\nabla \ln \pi_\theta$ is equal to [15]

$$\nabla \ln \pi_\theta = \frac{\nabla \pi(a_t|s_t,\ \theta)}{\pi(a_t|s_t,\ \theta)} \tag{16}$$

The intuition behind this formulation is that the policy parameter vector θ is shifted in the direction of the steepest ascent with positive return G, to improve the policy π_θ after each step.

REINFORCE is a valuable policy search method that creates a state value function and has guaranteed convergence; however, it tends to converge slowly compared to other bootstrapped methods.

3.2.2. Actor-Critic Methods

The one-step advantage actor-critic (A2C) is analogous to the TD methods introduced above, which do not involve offline policy improvement steps. A2C implements the notion of δ_t, the difference between the expected value and actual value for a given state and state-value weighting w [15].

$$\delta_t = R_{t+1} + \gamma \hat{v}(s_{t+1},\ w) - \hat{v}(s_t,\ w) \tag{17}$$

The difference parameter can be used each step to update the state-value weighting w and the policy parameter θ with the equation [15].

$$w \leftarrow w + \alpha^w \delta\ \nabla \hat{v}(s|w) \tag{18}$$

$$\theta \leftarrow \theta + \alpha^\theta \delta\ \nabla \ln \pi_\theta(a|s,\theta) \tag{19}$$

The calculation for the policy and difference parameter is completed after each incremental exploration step in the search space.

Another common actor-critic approach seen in the literature is asynchronous advantage actor-critic (A3C) [29]. A3C has the same formulation as A2C, with the slight difference of allowing multiple agents to asynchronously explore different policies in parallel. This approach reduces the need for experience replay to stabilize learning in time and improve stability.

3.2.3. Deterministic Policy Gradient

Another common implementation is the deterministic policy gradient (DPG) technique [30]. DPG implements Q-learning updates to determine the action value function as shown [30].

$$\delta_t = R_t + \gamma Q^w(s_{t+1},\ \mu_\theta(s_{t+1})) - Q^w(s_t, a_t) \tag{20}$$

In this approach, DPG moves the policy in the direction of the gradient of Q for continuous spaces, rather than selecting the globally optimal Q value in each step [30].

$$w \leftarrow w + \alpha^w \delta\ \nabla_w Q^w(s_t, a_t) \tag{21}$$

$$\theta \leftarrow \theta + \alpha^\theta \nabla_\theta \mu_\theta(s_t)\ \nabla_a Q^w(s_t, a_t)\Big|_{a=\mu_\theta(s)} \tag{22}$$

Here $Q^w(s_t, a_t)$ is a differentiable action value function which is used to replace the true action value function.

3.2.4. Proximal Policy Optimization

Proximal policy optimization (PPO) is a policy gradient technique which was designed to provide faster policy updates then A2C or DPG. PPO applies the DPG structure, but updates the policy parameter θ based on a simple surrogate objective function [31]. The policy update includes a step of minimizing the penalty term associated with the difference between the surrogate and the original function.

3.3. Summary

The methods reviewed are fundamental to the different implementations of RL in robotics. The choice in policy optimization has a significant impact on the convergence guarantees and training time required.

4. Reward Shaping

In RL, the allocation of rewards drives the training process because the agent aims to maximize the reward signal [32]. Reward shaping is the technique that involves modifying the reward function to help the agent learn faster [33]. An efficient reward function can accelerate the learning process by introducing background knowledge to RL agents [34].

Traditional RL applications used monolithic goals exclusively [35]. Monolithic goals seemed to behave perfectly with well-behaved domains with clear reward objectives. However, as RL problems started to become more complex in the 1990s, this approach was found to be inadequate for modeling real physical world situations (such as robotics). The difficulties in recognizing the state information, nondeterministic and noisy environment, variances in the learning trails required, timeliness of the rewards, and the requirement of achieving multiple goals can contribute to the failure of monolithic goals for these applications [35].

To address these challenges, as an improvement to a naïve monolithic single goal reward function, Mataric proposed a heterogeneous reinforcement function [35]. The method broke the rewards down into several goals representing the knowledge of the domains, then grouped these goals with progress estimators. Goal refinement enables the system to learn through immediate transitionary rewards which can speed up learning and convergence. The issue with this approach is that the agent can achieve a net positive gain in reward by running cycles rather than finding the end state [21,32].

To address this, Ng et al. [21] introduced a potential-based shaping algorithm. The algorithm incorporated a new reward function: $R' = R + F$, where R is the traditional reward and F is the shaping reward function. The shaping reward function F is a function over states, which should be chosen or constructed [34] based on the domain knowledge between two states. For example, F could be a function of distance to the goal state, wherein F increases as the agent selects actions closer to the goal state. Ng et al. proved that the optimal policy would not change given this new reward function [21].

Adding rewards is advantageous when the rewards represent the actual effect of the actions, however, adding rewards can also cause errors in the learning process when noisy reward inputs are given. To find balance between efficiency and effectiveness in RL reward shaping, Luo et al. proposed a method called Dense2Sparse [36]. By letting the system receive rewards continuously in the first several learning episodes, the policy is expected to converge quickly despite the noisy information. The suboptimal policy learned from the dense reward can be optimized under a noise-free sparse rewarding scheme, where a positive reward is given only when the task or sub-task is completed. Jang et al. found the concept of combining sparse and dense reward schemes to be effective with their research on training a Walker robot with high dimensional continuous action and state spaces [37].

Besides using a predefined change in the reward function, reward shaping can also be performed dynamically throughout the learning process. Tenorio-González et al. realized a dynamic reward shaping technique by implementing verbal feedback from a human observer [38]. The inclusion of human intervention can help the system converge faster but requires active human participation in the learning process. Konidaris and Barto [39] developed a method to improve the reward function autonomously. Their work focused on generalizing the knowledge learned by the agents for small tasks to produce a shaping function to apply to all rewards. Their experiment showed good performance of the shaped rewards in a rod positioning scenario.

Reward shaping is a tradeoff between accuracy and speed. A monolithic goal reward can be a noise-free input, but it is sometimes not practical for complex tasks. Breaking

down the task and applying rewards frequently allows the system to learn faster, but it can also distract the agent [32,40].

5. Imitation (Apprenticeship) Learning

Reward shaping is one of the tasks which require human intervention in RL. With only a monolithic goal, the agent may be forced to search the environment for an extended period to refine its actions. An alternative path to manual reward shaping is imitation learning [15,41,42].

Traditional robotic control systems (discussed in Section 1.1) can have imitation behavior programmed into a robot via manual robotic arm positioning. This approach has the equivalent results to implementing the teach pendant without the need for joint programming. Manual arm positioning is valid but has all the same issues as traditional control, i.e., the inability to handle changes in the environment [7,8].

As an alternative to traditional control, imitation learning can be used to learn a reward function from examples. The reward function can be used to refine the policy depending on the scenario, allowing for domain adaptation [42]. Imitation (apprenticeship) learning includes several different approaches, such as Behavior Cloning (BC) and Inverse Reinforcement Learning (IRL). Like all other approaches to RL, imitation learning has the option to implement model-based and model-free techniques [42].

5.1. Behavior Cloning

BC is the simplest form of imitation learning, as it borders supervised learning (SL) and RL. The objective of this learning technique is to learn the policy which determined the trajectory of the expert demonstration. The trajectory variable $\tau = (s_0, a_0, s_1, a_1, \ldots)$ is used as the functional output from the policy [42]

$$\tau = \pi^*(s) \tag{23}$$

where s is the state of some robotic manipulator. The problem resembles supervised learning, where the objective of mimicking behavior is set as minimizing the loss between the policy parameters θ of the policy $\pi_\theta(s)$ and the optimal policy based on expert demonstrations of $\pi^*(s)$.

BC is easy to understand and execute, but since the problem is reduced to a supervised learning approach, error may occur when reaching a state which the expert has never visited. The policy may attempt to map a new environment with the same strategy used for an older environment, which can cause an error referred to as the covariate shift [43]. Due to undefined behavior in that state, the algorithm could lead to catastrophic problems, as was shown by Stephane Ross in his work on mismatch training and test inputs on training robotics [44].

The first application of behavior cloning was ALVINN, a project by Pomerleau in 1989 [45], where a neural network was used to estimate the policy for driving an autonomous vehicle. The inputs were sensor signals from vision cameras and laser range sensors, and the outputs were turning curvatures which translated into steering angles. Supervised learning was performed by using road snapshots and turn curvature pairs to train the network. The objective suggests that it minimizes the 1-step deviation error along the expert trajectory; therefore, it does not capture the intention of expert behavior in the long term, which reduces the accuracy of long-term planning.

Due to the approach operating in the SL domain, BC is more applicable to scenarios in which massive databases of expert data are available [43]. BC has been shown to be valuable in large data domains such as autonomous vehicles [46] but has not been shown to be applicable to robotics.

5.2. Inverse Reinforcement Learning

IRL is a way to find the underlying reward function which explains the agent's behavior (the policy). IRL is based on the known parameters in an MDP $(s, a, p(s'|s, a), \gamma,$

π^*) [41,42]. The goal in IRL is to choose R, which makes the policy π optimal, and makes any changes in policy π as costly as possible, so that the differences between two actions in any state is significant [41].

The notation for IRL is [47]

$$R^*(s) = w^* \phi(s) \tag{24}$$

where the true reward function R^* is a function of features ϕ, and the true weights w^*. In robotics, ϕ could be used to represent the features: smooth motion profiles and speed. Different weights w^* could represent the tradeoff between these two features.

One approach for implementing IRL is with the apprenticeship learning technique implemented by Abbeel and Ng [47]. This technique defines the value of a policy as:

$$E_{s_0 \sim\ p(s'|s,a_1)}[V^{\pi}(s_0)] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \middle| \pi\right] = w * E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \middle| \pi\right] \tag{25}$$

where $E_{s_0 \sim p(s'|s,a)}[V^{\pi}(s_0)]$ represents the expectation for value given that the agent starts in state s_0 with some transition probability for the next state of $p(s'|\ s,a)$ [41]. To get the final formulation shown in Equation (25), the value function for the policy is replaced with the reward function defined in Equation (24). This formulation can be used to find the policy $\widetilde{\pi}$ by minimizing the difference between the policies π_E and $\widetilde{\pi}$ while maximizing the minimum differences between trajectories. The formulation for this min-max problem can be expressed as [47].

$$\epsilon \geq \max_w \min_j [w^T \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \middle| \pi_E\right] - \left[\sum_{t=0}^{\infty} \gamma^t \phi\left(s_t^k\right) \middle| \widetilde{\pi}\right]] \tag{26}$$

The IRL learning process involves the loop shown in Figure 3, where the policy is initially randomized. First, Equation (26) is implemented to compare the policies of the expert and the agent and improve the reward weighting w^T. Next, the updated reward weighting is used to calculate the state rewards (Equation (24)). Finally, the RL algorithm is used to compute the optimal policy.

Figure 3. IRL learning process.

The IRL solution is often ill-posed, so instead of forcing the system to perform exactly as the optimal behavior from the demonstrations, the constraints can be relaxed to help find a suitable solution. Abbeel and Ng [47] did so by relaxing the equality of the performance with

$$\max_{\pi \in \Pi} \mathbb{E}_{\pi}[r(s,a)] \geq \mathbb{E}_{\pi^*}[r^*(s,a)] - \epsilon \tag{27}$$

Algorithms based on IRL can be costly due to the inner loop, which contains internal RL calculations for each step. The reward function acquired with IRL does not tell the agent how to act, thus imitating the exact motion of the expert demonstrations still requires RL [43]. That said, IRL has shown significant potential in robotics applications, as will be shown in the Section 8.

6. Pose Estimation for Grasp Selection

Pose estimation is a critical factor for pick and handle operations. Incorrect estimation of pose could result in poor grasp choice due to the robotic arm pose or unstable placement due to incorrect estimation of object pose. Gualtieri and Platt [19] noted the value in pose estimation when comparing the accuracy of cup placement against blocks and bottles. Blocks are symmetric in x, y, and z, bottles are symmetric in x and y, and cups are symmetric only in x. The complexity of the cup pose made the pick-and-place task more difficult, which resulted in lower overall accuracy compared to blocks and bottles (Figure 4). Pose estimation is critical, as the task should be completable regardless of the complexity of the target object.

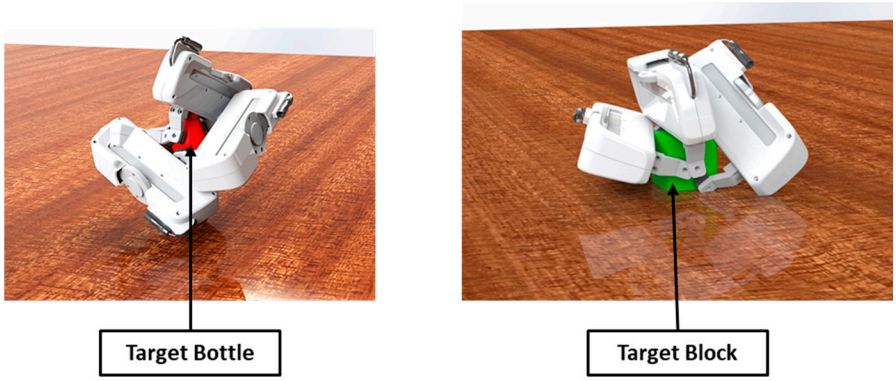

Figure 4. Various grasp pose configurations for bottle and block targets. Bottle targets are symmetric in x and y compared to blocks which are symmetric in x, y, and z.

The literature refers to two pose types, the object pose and the gripper pose (also referred to as the grasps [48]). The object pose is described by the position and orientation of the object being manipulated [49,50]. Each object pose exists inside the set of all possible poses for that object, called the pose space [50]. The gripper pose is used to describe the different gripping positions, orientations, and jaw openings that could be used to grasp a target object [23]. Each gripper pose exists inside the set of all possible grasps for an object, called the grasp space. Pose estimation can be completed using analytic (geometric) or data-driven methods.

Analytic methods fit the point cloud of sensor input to known models (often CAD). Model fitting limits the agent to detecting the pose of items which have models available for comparison [51]. The analytic method first determines the target object pose, then selects an optimal gripper grasp based on the target object pose. The analytical method is usually set up as an optimization problem. The grasp is chosen from the grasp space based on maximizing the values for resistance to disturbance, dexterity for further manipulation, equilibrium to reduce object forces, and stability in the case of external forces or grasp errors [52,53]. The analytic method is useful when precise placement in a particular orientation is required [23].

Data-driven methods are free from the constraint of requiring models, as this approach implements machine learning to estimate optimal gripper poses directly [23]. There are two approaches inside of the data-driven method: the target-model-based approach and the model-free approach.

The target model-based approach implements the point cloud data from the sensors to estimate a model of the target object. The target model can be used to eliminate many of the available grasp pose options. Supervised learning (SL) can be used to label objects based on the training data [23,53]. To save time on data labeling for the SL problem, simulation environments can be implemented to create a database of randomly oriented and mixed samples for training the model. The data-driven target model-based approach is optimal when accurate placement and correct target orientation is critical to receive the reward or when the target objects must be selected from a cluttered scene [23].

The model-free approach removes the model estimation step and instead focuses on optimizing the grasp pose based on the entire range of available grasps. The model-free approach is composed of discriminative and generative techniques. The discriminative technique reviews all the available grasp pose candidates and trains a convolutional neural network (NN) to select the optimal grasp based on these [19]. The generative approach recommends a grasp configuration based on the orientation of the target. This approach is fundamentally the same as object detection, where the gripper pose is selected based on the orientation of the detected object. An example of this comes from Jiang et al. [54], who implemented orientation rectangles to provide an estimate for the ideal position and orientation of the gripper head. Since multiple grasp candidates are often available, a NN is used to train the model to select the best grasp. The model-free approach is most successful when dealing with new targets and soft placement restrictions [23].

The approaches listed work well in combination with other strategies. For example, the agent could be allowed to implement the temporary placement of the object, free from clutter, to allow a better estimation of pose. This approach would be useful when accurate placement is required [19]. An alternative method would be to implement multiple viewing angles by hoisting the item in front of external cameras. Multiple viewing angles could be used to identify the object or determine if a more optimal grasp pose is available [55]. Another practical approach for estimating pose is by implementing focus. Attention focus has been successfully implemented by Gualtieri and Platt [22] via iterative zooming to regions of interest. The use of focus would limit the number of available grasp poses to those seen in the focus region.

7. Simulation Environment

In the sections above, training in simulation was mentioned as a requirement for some approaches, and a useful tool for others. The benefit of training in a simulated environment is that the agent can perform thousands of training iterations in a short period, without any wear effects on the online device. Online training requires the availability of the robot and human supervision for safety reasons. Examples of the difficulties with training in real life can be seen from Levine and Koltun [28], who implemented 14 robotic manipulators over two months to collect 800,000 grasp attempts. The accuracy of the grasp operation after this training was 82.5 percent, a lower accuracy than has been achieved for comparable pick-and-place actions when training was completed in simulation.

The benefits of training offline are often contrasted with the loss of accuracy when the trained model is applied in the real world. Inaccuracies of model parameters such as friction, weight, or dimension, may have massive influences during model testing. The difference between the offline and online models is referred to as the "reality gap". The primary approaches for coping with the reality gap are domain randomization and domain adaptation [23].

Domain randomization is a method of training the agent in an inconsistent environment. The goal is to force the agent to learn to ignore "noise" and changes in the background. Noise added for domain randomization could include lighting, color, positions, textures, or other features that serve to confuse the scene. The goal of domain randomization is to make the online test scene appear as a training sample with different randomizations than previously experienced [56].

Domain adaptation is an approach that implements a form of generative adversarial networks (GANs) to generalize the test data onto the target domain. The process implements data from the test domain (online) to train the model in the training domain (offline) so that training in the simulation generalizes well to online testing. This approach still requires a small amount of labeled data from the test domain; however, much less data is required than online training [23,57].

The choice of simulation environment is another factor that affects test accuracy. An assortment of packages exist (MuJouCo, ODE, and Bullet), each of which have their pros and cons. Key factors include consistency, stability, numerical accuracy of simulation, and

compatibility [58]. MuJouCo performs better than other simulators in terms of simulation accuracy, energy conservation, and grasp stability. ODE came in second for nearly all of these measurements. The primary difference between the two physics engines is that ODE is open source and compatible with a variety of simulators such as Gazebo and OpenRAVE [59]. In terms of support, documentation, and ease of use, MuJouCo was ranked the worst [58].

8. Analysis

The above sections review the primary choices that need to be made during problem formulation. This analysis section focuses on reviewing and critically discussing the current state of research.

8.1. State of Research—Complete Pick-and-Place Task

Benchmarks for pick-and-place operations include simulation accuracy, the number of online tests required to train the model, and most importantly, the accuracy of testing. The table below reviews papers published by leading RL robotics researchers to illustrate the strengths and weaknesses of the different implementations. Success rates in each paper are difficult to compare due to the differences in the tasks. The discussion outside of the table gives clarity for each implementation and explains the results.

Fu et al. [17] completed several tasks, with comparable difficulty to pick-and-place, including inserting a peg in a hole, stacking toy blocks, placing a ring on a peg, and more. The approach involved learning the system dynamics from prior examples, then completing manipulation tasks in one shot after the goal was defined. The approach did not directly use RL for learning individual tasks by searching through the state space, but instead used RL to learn the system dynamics for robotic control. The accuracy of these tasks was high; however, the approach did not incorporate a cluttered scene.

Gualtieri et al. [19,22] published several papers on completing the pick-and-place action in a cluttered scene with an assortment of objects. Before this work, Gualtieri et al. [51] first published a manuscript on pose estimation. The pose estimation research completed in [51] was applied to the entire pick-and-place operation in [19,22]. The group completed testing in simulation and online. The accuracy listed represents the online accuracy of grasping a novel object after being trained on objects of similar type. The group found that an item picked from a clutter of objects had significantly lower accuracy than selecting an object in isolation. The results indicate that items with complex geometry (cups and bottles) are significantly more difficult to grasp compared to targets with a simple geometry (square blocks). Compared to other tasks completed in the literature, the complexity of the cluttered pick-and-place task was high.

Popov et al. [60] completed the action of precision stacking of Lego blocks using distributed deep DPG (DDPG) with a robotic arm in simulation. The blocks were initially positioned in isolation in the same orientation, which significantly reduced the task difficulty. The team compared simple single reward and composite multi-step reward functions and proved that the single reward approach had the fastest convergence. The groups asynchronous implementation of DDPG proved that distributed training with asynchronous updates can improve convergence speed. The task had an average complexity.

Mahler and Goldberg [61] applied imitation learning to complete the task of selecting novel objects from a cluttered environment and placing them aside in an organized manner. The learning approach implemented the Dex-Net 2.0 database of point clouds and grasp poses to generate training samples for learning effective Grasp Quality Convolutional Neural Networks (GQ-CNN). The policy learned appropriate NN weights based on replicating the results from the demonstrated samples. Transfer learning was effectively used to apply synthetic training to real world pick-and-place. Compared to the other examples from literature, the complexity of this cluttered pick-and-place task was high.

Sehgal et al. [62] simulated a simple, single block pick-and-place task with the use of DDPG HER whose parameters were tuned with the genetic algorithm. The main

contribution from this paper was to show that the GA could be used to minimize the number of training iterations required to achieve comparable convergence to that of DDPG HER without the GA. The overall task complexity for this action was low. Zuo et al. [63] implemented deterministic GAIL to complete a simple, single block pick-and-place task. The DGAIL approach implemented a NN discriminator to develop a reward function based on examples and a policy gradient method for learning ideal actions based on the discriminator. During testing, DGAIL was compared to stochastic GAIL and DDPG to prove the relevance of this approach. The approach yielded much better results than all other methods except DDPG, a more computationally expensive approach. The overall task complexity for this implementation was low.

Chen et al. [64] completed a picking operation where the target object was in a cluttered environment. The robotic arm could rearrange the cluttered scene if the target was invisible from the camera vantage. The approach was limited due to the model-based grasp pose technique, however, it showed excellent performance in finding the target object in the cluttered scenario. The placement operation was not tested but could have been easily implemented since models for the target object were available. The action of selecting the target from a cluttered scene had an average complexity.

Xiao et al. [65] achieved a high accuracy for the pick-and-place task in a cluttered environment with the Parameterized Action Partially Observable Monte-Carlo Planning (PA-POMCP). The system approximated the utility of available actions based on the current belief of the agent about the environment. The approach allowed the agent to change perspective or maneuver items in the scene to find an accurate estimation for the location of the target object. The approach showed significant performance for known items but was not tested on unknown items in new environments. The action sequence of rearranging the scene and selecting a target from the clutter, had a high complexity.

Liu et al. [66] designed a system to complete several robotic pick-and-place tasks, including stacking, sorting, and pin-in-hole placement. A major contribution of this paper was the work around reward shaping to solve complex problems. The approach assigned rewards based on the Euclidean distance between the Objects Target Configuration (OTC) and the Objects Current Configuration (OCM). The closer the configuration of blocks, pins, etc., to the optimal configuration, the more reward allocated to the sample. Additional rewards were allocated for final task completion. Comparison between this approach and a basic linear reward approach shows a significant improvement in convergence. In addition to reward shaping, this paper compared the MAPPO learning technique to traditional PPO and Actor-Critic (A3C) techniques. MAPPO improved performance against the other techniques by directly outputting manipulation signal to the agent. The overall task complexity for the various action sequences was high.

Mohammed et al. [67] simulated a robotic agent which completed the task of organizing 10 blocks dropped into a workspace. The training technique applied was Q-learning. The block geometries were not all the same, but the shapes were simple. The accuracies for the pick-and-place operations were only tested in simulation, and not applied to an online agent. Considering the average-high complexity of the task, the accuracy of the pick-and-place and the computational efficiencies are promising.

Li et al. [68] applied a robotic arm to perform tasks including reach, push, multi-step push and pick-and-place using the Augmented Curiosity-Driven Experience Replay (ACDER) approach. The pick-and-place operation was applied to a single target object which was free from clutter. The novelty of the research includes a method of exploration in which rewards are allocated for the exploration of unfamiliar states (curiosity). Additionally, this approach initiated the robot in states not stored in the replay buffer to improve exploration efficiency. Through testing, this approach was proven to be several times more sample efficient than hindsight experience replay (HER). The overall task complexity for this action was low.

Pore and Aragon-Camarasa [69] proposed an algorithm for robotic pick-and-place on a simple block placed in isolation on a surface. The solution method involved decomposing

the task into *approach*, *grasp* and *retract* segments. After perfect accuracy was achieved for each independently trained action, a high-level choreographer (actor-critic network) was trained to learn the policy for ordering the behaviors. The team compared a DDPG + HER approach to this A3C + subsumption architecture (SA) model. The end-to-end approach with DDPG + HER resulted in a maximum of 60% accuracy after 10,000 episodes. The A3C + SA model achieved 100% accuracy after 6000 training episodes. The relative comparison shows the potential of this SA model; however, the task had a low complexity and was never tested online with a real robot.

Al-Selwi et al. [70] simulated the pick-and-place task on a simple block placed in isolation on a surface using a DDPG approach combined with HER. The approach validated what has been shown in other papers [19], that target object complexity changes task effectiveness. The work presents little novelty in terms of policy optimization. The overall task complexity in this example was low.

Marzari et al. [71] presented simulations and online testing for pick-and-place task on a simple block placed in isolation on a surface by using a DDPG HER method. By implementing task decomposition, the model was able to achieve 100% accurate performance with simulation and online testing. The major contribution of this approach compared to [69] was that behavior cloning was not used for training the *approach* subtask, so sample operations were not required. Online testing indicated excellent generalizability for novel objects and presented a high sample efficiency compared to all other approaches. Although the overall task complexity was low, this approach shows excellent potential for future applications.

Anca and Studley [72] completed a simulated pick-and-place operation in which a simple block was picked by manipulating a robot in 2D with a twin delayed actor-critic network. The result demonstrated operational accuracy without requiring example simulations (unlike [69]), however, showed little advantage over approaches based on DDPG HER [71]. The task complexity with this action sequence was lower than all other samples.

8.2. State of Research—Pick-and-Place Subtasks

Due to the novelty of the field, Table 1 presents the (short) exhaustive list of all robotic pick-and-place with RL literature currently available. That said, there are other publications which present contribution to the field, that do not include key elements of the action sequence. Examples include the literature on grasp selection [73,74] or precision placement [75]. To provide a comprehensive survey, key examples for the pick-and-place subtasks drawn from the literature are listed in Table 2.

Finn et al. [75] developed an IRL algorithm that could be applied to various tasks in different environments to learn the policy and reward function. A task completed in simulation included peg insertion in a hole to a depth of 0.1m. The robotic model could complete this task with high precision after training with ~30 examples. Of the real robotic tasks completed, the task of placing dishes in a rack was the most like pick-and-place. After 25–30 samples of human teaching, the dish placement action could be completed with perfect accuracy. This research gives an excellent indication of the power of IRL, however, to apply this algorithm to the full pick-and-place action sequence, further extension is required to include the pose estimation step for "picking".

Kalashnikov et al. [74] developed a custom Q-learning approach focused on generalizability and scalability for unfamiliar environments. The task required the picking of random objects from a dense clutter and raising the object above the clutter. The training was completed online to incorporate all properties which were unmodeled in the physics simulation. To apply this algorithm to the full pick-and-place action sequence, the research requires additional task programming to include precision placement.

Table 1. Summary of RL in Robotic Pick-and-place.

Paper	Policy Optimization	Pose Estimation	Sim Package	Success Rate	Strengths (S) and Weaknesses (W)
Fu et al., 2016 [17]	Value function approach with DP using iterative LQR. Basic two-layer NN for updating priors	Model-based approach which combined prior knowledge and online adaptation	MuJoCo	Simulation Testing: 0.80–1.00 Robot Testing: ~0.80	S: Implemented prior knowledge to reduce the number of samples required W: To create model of system dynamics the NN must be trained on prior examples
Gualtieri et al., 2018 [22], Gualtieri et al., 2018 [19]	Value function approach with SARSA. Caffe (NN) used to update the SARSA weights	Model-free data-driven approach which implemented focus. The algorithm allowed for temporary and final placements. Model was trained using 3DNET CAD models	OpenRave (ODE base)	Robot Testing: Picking: 0.88–0.96 Placement: 0.80–0.89	S: Model-free approach with high picking accuracy W: Low placement accuracy and low overall accuracy in testing
Popov et al., 2017 [60]	Distributed Deep DPG (DDPG) method with efficient schedule updates and composite reward functions	Model-free data-driven approach. Reward achieved for appropriate grasp selections. Blocks always oriented in the same manner.	MuJoCo	Simulation Testing: 0.955	S: Proved that asychronous DDPG has higher data-efficiency then other approaches W: No online testing to validate simulation accuracy
Mahler and Goldberg, 2017 [61]	Imitation learning approach in which the robot was trained on data from synthesized grasps. Sample grasps were found by using force and torque space analysis and knowledge of object shapes and poses	Model based approach used for demonstration synthesis. Model free data driven approach for pose selection during training	Pybullet	Simulated Testing: 0.96 Robot Testing: 0.94–0.78 (Testing accuracy depends on the number of objects in the cluttered environment)	S: High picking accuracy on cluttered enviroment of unfamilier objects W: Requires models for the wrench space analysis to develop demonstration samples. Intensive programming effort
Sehgal et al., 2018 [62]	DDPG with hindsight experience replay (HER). Parameters tuned using the genetic algorithm	Model-free data-driven approach	MuJoCo	Simulated Testing: ~0.90	S: Effectively implemented HER for faster convergence W: No improvement in overall task completion performance. No online testing to validate simulation accuracy
Zuo et al., 2019 [63]	Deterministic generative adversarial learning (DGAIL) which implemented a policy gradient generator and discriminator trained with IRL (an actor-critic approach)	Data driven approach in which action selection was based on the difference between the demonstrated and generated policy	MuJoCo	Simulated Testing: 0.90	S: DGAIL had faster convergence then several other RL techniques W: DGAIL was less stable, and less accurate then other modified DDPG techniques

Table 1. *Cont.*

Paper	Policy Optimization	Pose Estimation	Sim Package	Success Rate	Strengths (S) and Weaknesses (W)
Chen et al., 2019 [64]	Two deep Q-Networks (DQN) for pushing and grasping. Mask Region convolutional NN (R-CNN) for object detection	Model based approach where the target object was mixed in clutter. RBGD image was used for target object detection. The scene was rearranged if the target is invisible from the sensor's perspective	V-REP simulation platform	Simulated Testing: 1.0 3 different scene options were tested. Each seen had the target in a more/less hidden location	S: High target location accuracy. Approach proves that rearranging the enviroment can improve results W: Approach was only applied for known target objects. No online testing to validate the simulation accuracy
Xiao et al., 2019 [65]	Parameterized Action Partially Observable Monte-Carlo Planning (PA-POMCP)	Model based approach which implemented known models from a benchmark model set	OpenRAVE & Gazebo	Robot Testing: 1.0	S: High pick-and-place accuracy W: Model based approach. Grasp poses are pre-defined so the primary task is model recognition. This technique significantly simplifies the problem
Liu et al., 2020 [66]	PPO with actor output as manipulation signal (MAPPO)	Data driven approach which extracted target object pose from RGBD sensor	Gazebo	Robotic Testing: Relative improvement of MAPPO to PPO and A3C shown to be >30%. Final accuracy not given	S: Compared various learning and reward shaping approaches W: No final pick-and-place accuracy given. Difficult to guage the performance improvement
Mohammed et al., 2020 [67]	Value Function approach with Q-Learning. CNN used to update Q-learning weights. Model pre-trained on Densnet-121.	Data-driven approach. Grasps were generated by using a CNN to identify available poses from a RGB image	Vrep (ODE base)	Simulated Testing: Picking: 0.8–1 Placement: 0.9–1	S: High placement accuracy considering the absense of model for target object. Short training time W: No online testing to validate the simulation accuracy
Li et al., 2020 [68]	DDPG approach with goal-oriented and curiosity driven exploration and dynamic initial states	Data driven approach which used RGBD images to determine object and goal positions	MuJoCo	Simulated Testing: 0.95 Robotic Testing: 0.85	S: Comparison to several other sampling and learning techniques validated the sample efficiency for this approach W: Deployment on robot showed a reduced overall effectiveness

Table 1. *Cont.*

Paper	Policy Optimization	Pose Estimation	Sim Package	Success Rate	Strengths (S) and Weaknesses (W)
Pore and Aragon-Camarasa, 2020 [69]	Hierarchical RL in which the task was broken into simplified multi-step behaviors with the subsumption architecture (SA). Behavior cloning was used for low-level behavior training. Actor-Critic technique was applied for high level task completion	Data driven approach in which grasp poses were trained with behavior cloning. Target objects were consistent which means that advanced pose estimation was not required	OpenAI Fetch environment with MuJoCo	Simulated Testing: 1.00	S: Validated that the subsumption architecture improves entire task performance significantly compared to end-to-end approaches W: Technique required behavior cloning samples, and a human for problem breakdown. No online testing to validate simulation accuracy
Al-Selwi et al., 2021 [70]	DDPG with HER	Model based approach in which RGB image was used to determine bounding box and rotation angle	MuJoCo	Simulated Testing: 0.502–0.98 (depending on target geometry)	S: Validated that HER DDPG can be used with vision feedback to improve accuracy W: CAD models required for pose selection. No online testing to validate simulation accuracy
Marzari et al., 2021 [71]	DDPG with HER and task decomposition	Data driven approach in which grasp poses were learned from a single target object with simple geometry	MuJoCo	Simulation and Robotic Testing: 100%	S: Proved that the DDPG approach with HER can perform excellent task completion accuracy by using task decomposition rather then end-to-end training W: Approach assumed human involvement for task decomposition
Anca and Studley, 2021 [72]	Twin delayed hierarchical actor critic (TDHAC) which broke task into high- and low-level goals	Data driven approach which only implemented a 2D motion for picking the action	Pybullet	Simulated Testing: 100%	S: Confirmed that the hierarchical approach improves convergence W: Approach showed no significant improvement over DDPG with HER. The motion was limited to 2D. No online testing to validate simulation accuracy

Table 2. Robotic Pick and Place Subtasks.

Paper	Policy Optimization	Pose Estimation	Sim Package	Success Rate	Strengths (S) and Weaknesses (W)
Finn et al., 2016 [75]	Combined relative entropy IRL and path integral IRL with sampling using policy optimization	Algorithm did not incorporate the picking operation so pose selection was not required	MuJoCo	Robotic Testing: Placement: 1	S: Perfect placement accuracy W: Technique did not incorporate picking action. Online training samples are required
Kalashnikov et al., 2018 [74]	QT-Opt, a modified Q-learning algorithm that implemented off-policy training and on-policy fine-tuning	Data-driven approach. Strategy implemented dynamic closed loop control with RL to solve the grasping task.	Bullet Physics Simulator	Robotic Testing: Picking: 0.96 (with object shifting)	S: High picking accuracy in a cluttered environment with unknown objects W: Time-consuming online training required
Wu et al., 2019 [76]	Pixel attentive PPO	Data driven approach which learned optimal grasps through trial and error. Pixel attentive method cropped image to focus on local region containing ideal grasp candidates	Pybullet	Simulation and Robotic Testing: 0.911–0.967. Accuracy changed based on the density of the clutter	S: Excellent alignment between simulation and real enviroments. Useful data presented which compared camera orientation and grasp accuracy W:Three finger gripper may contribute to high picking success rate. Most standard grippers are planar (2 fingers)
Deng et al., 2019 [77]	Deep Q-Network (DQN) implemented to select actions based on affordance map	Data driven model-free approach in which lifting point candidates were selected based on affordance map showing "hot spots" or ideal grasp candidate locations in the RGBD image.	V-REP	Robotic Testing: Picking: ~0.71	S: Novel robotic arm design effective for selection of randomly oriented objects in a clutter. W: Picking operation success rate was only 11% better then random grasp actions (very poor)
Beltran-Hernandez et al., 2019 [78]	Guided policy search, with image input and grasp pose as output.	Model-based approach. The agent was trained on basic shapes and then tested on complex geometries	Gazebo (ODE Base)	Simulation Testing: Picking: 0.8–1	S: Approach shows significant controbution to the space by showing effectiveness of using model based techniques for grasping unfamiliar objects W: No online testing to validate simulation accuracy
Berscheid et al., 2019 [79]	Modified Q-learning. Upper confidence bound for off-policy exploration.	Model-free approach which implemented NN to generate action values	Online Testing. 25,000 grasps trained in 100 h	Robotic Testing: Picking: 0.92 (with object shifting)	S: High picking accuracy and good recognition of required shifts W: Time-consuming online training required
Kim et al., 2020 [80]	A2C with state representation learning (SLR). Involves learning a compressed state representation	Data driven approach which implemented raw image disentanglement	Pybullet	Simulation Testing: Picking: 0.72	S: Computationally affordable grasping action W: Low picking accuracy. No online testing to validate simulation accuracy

Wu et al. [76] designed a custom robot which implements a 3 finger gripper to pick objects out of a clutter of 2–30 objects. The study demonstrated that attention improved the picking action in cluttered scenes by ~20% and implementing a 3-finger gripper improved object selection by ~45%. To apply this technique to the full pick-and-place operation, additional task programming is required for precision placement.

Deng et al. [77] designed a custom robotic hand to be used for picking objects out of a cluttered scene with a series of fingers and a suction rod. The mode of selection allowed for a simple convolutional NN structure to be used for selecting ideal grasp locations. During testing, grasp failure occurred if 3 consecutive grasp attempts were not completed sequentially. Task simplicity and low-test accuracy limits this approach for real world application.

Beltran-Hernandez et al. [78] performed the task of picking simple geometry items (cylinder, cube, and sphere) during simulation training. Accuracy for simulation testing on random objects (duck, nut, mechanical part) was high, especially considering the major differences between the trained models and the test samples. The approach was model based, so it could be easily extended to incorporate high precision placement.

Berscheid et al. [79] completed the robotic task of manipulating and picking a variety of objects in a clutter with the application of Q-Learning. Results indicate high picking effectiveness and generalizability. Training was completed online, without the use of a simulation model. The task definition did not include placement, but the picking action included a high level of difficulty. To apply this algorithm to the full pick-and-place action sequence, additional task programming is required to include precision placement.

Kim et al. [80] completed the task of picking up unknown target objects in a highly cluttered scene by using disentanglement of image input. The goal of this approach was to reduce the computational difficulty of performing a pick action. A key contribution of this paper is showing the value in creating a low dimensional state representation of the target object with the use of autoencoders. Additionally, this paper investigated the value of implementing disentanglement to allow simple scene recognition to guide behavior. The disentanglement approaches include attention, separation of internal (robot) and external (scene) information, and separation of position and appearance in the scene.

Many other papers exist, which can be applicable for subtasks of pick-and-place, however most of them could not be easily extended to the entire operation. For more samples of individual grasping techniques for the pick action, this author recommends the review of [12].

8.3. Critical Discussion

The literature discussed above shows that significant progress has been made towards developing a RL approach which can be applied to robotics, however there are several issues with each approach. Additional research is required for validating simulations, reducing the loss in accuracy from crossing the reality gap, testing task variations, and lowering the amount of human intervention required.

Only 40% of the literature analyzed above includes results for online testing. Most of the research only included data on testing the models in simulation, wherein the dynamics are completely dependent on accurate modeling. To validate that the various methods are accurate and implementable in real life, further testing is required.

Of the manuscripts examined, several had robotic testing accuracies which were significantly lower than simulated accuracies [17,68]. Deviation between simulated and testing accuracies indicate that problems with bridging the reality gap require further investigation. As previously noted, training robotic agents online requires extensive time and equipment availability [28]. For RL to be applicable for robotic pick-and-place in industry, simulations must be developed in a way that allows for knowledge learned in simulation to be accurately transfer to the real world.

The literature includes many examples where the task involved simple pick-and-place with blocks or spheres in an isolated scene [62,63,68–72]. This type of task is rare in an

industrial environment, as most tasks involve a cluttered scene with specific placement requirements. Additional real and simulated testing is required to validate the accuracy of these approaches for more realistic tasks.

The methodologies which achieved robotic test accuracies high enough to be considered for application in industry, such as [61,71] required extensive human intervention. Task decomposition significantly speeds up learning and improves overall accuracy, however this method requires human involvement. The research must be extended to include methods for automatic task decomposition, or to develop strategies for task decomposition which do not require significant effort.

9. Open Problems

RL with robotics is a young and growing field which has many open problems to consider. The key problems include the lack of generalizability, the fostering of curiosity, and pose selection.

Problem generalization is an open topic, which relates to the issue of applying learning from the training set to a broader spectrum of problems. Ref. [78] gave an excellent example of a technique that can scale a crude model to a large test set, however, this approach still requires online testing.

Another quandary to be solved is finding a method to teach the agent to learn independently with intrinsic motivation. Sutton and Barto describe this idea as "curiosity" [15]. The goal is to enable the agent to learn tasks that may be useful in the future on its own, in order that the robotic arm could be more universally applicable.

Grasp pose selection is another open problem which significantly affects the pick-and-place task. The literature presented samples of pose selection which implemented grasp search or model recognition. Grasp search approaches proved to be reasonably effective when combined with focus, iterative zooming, or scene rearrangement [55,77], however this approach had mixed results when trained end-to-end. Model-based approaches generically performed well, however this approach was limited to tasks involving familiar target objects.

10. Conclusions and Future Work

This literature survey has shown that RL has potential to replace traditional robotic control. Several implementations show promising results for enabling robotic arms to perform basic pick-and-place actions consistently with high accuracy. The existing literature is optimistic; however, more research must be completed for open problems such as problem generalization, independent learning and grasp pose search.

To improve task generalizability, training could be extended to include a large sample set. None of the research reviewed in this paper presented pick-and-place actions where the targets between each action were drastically different. Generalizing the training samples could make the pick-and-place task more widely applicable. The idea is conceptually equivalent to domain randomization in which the task is randomized rather than the environment.

Self-motivated learning could be developed for the robotic agents, by rewarding "play". The agent could be taught to extend their learning by completing a self-selected activity. The explorative "play" actions could be used to optimize other tasks that are currently completed in a specific way to maximize the reward function. Intrinsically motivated rewards could allow the agent to explore new actions for future use [15].

Grasp pose selection could be optimized by combining the model-based and model-free techniques for pose estimation. The agent could implement a large bank of models for application with familiar target objects. In circumstances where no model matches the target object with a high accuracy, the agent could instead implement a model-free approach. This combined model-free and model-based approach could be developed to replicate the human approach of identification. Humans identify objects by mapping them

onto previously seen samples. If no sample is available, a human creates a new model for the object based on the new input.

RL shows promise for implementation in the future. By applying cutting edge policy learning techniques, robotic pick-and-place tasks can be trained with minimal human intervention and high accuracy [71]. Because of the nature of the learning process, similar manipulation tasks with comparable difficulty could be trained with minor adjustments of reward functions. For the field to continue to grow, further research is required to address the open problems of generalizability, self-motivation, and grasp selection.

Author Contributions: Conceptualization, A.L.; methodology, A.L.; writing—original draft preparation, A.L.; writing—review and editing, Y.Q.; supervision, H.-J.K.; project administration, Y.Q.; funding acquisition, H.-J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was Korea Electrotechnology Research Institute (KERI), through Korea-Canada AI Research Program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chang, G.; Stone, W. An effective learning approach for industrial robot programming. In Proceedings of the 2013 ASEE Annual Conference & Exposition, Atlanta, Georgia, 23–26 June 2013.
2. Massa, D.; Callegari, M.; Cristalli, C. Manual Guidance for Industrial Robot Programming. *Ind. Robot Int. J.* **2015**, 457–465. [CrossRef]
3. Biggs, G.; MacDonald, B. Survey of robot programming systems. In Proceedings of the Australasian Conference on Robotics and Automation, Brisbane, Australia, 1–3 December 2003; p. 27.
4. Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2008.
5. Craig, J.J. *Introduction to Robotics Mechanics and Control*; Pearson Education International: Upper Saddle River, NJ, USA, 2005.
6. Hughes, C.; Hughes, T. *Robotic Programming: A Guide to Controlling Autonomous Robots*; Que: Indianapolis, IN, USA, 2016.
7. Kumar Saha, S. *Introduction to Robotics*, 2nd ed.; McGraw Hill Education: New Delhi, India, 2014.
8. Ajaykumar, G.; Steele, M.; Huang, C.-M. A Survey on End-User Robot Programming. *arXiv* **2021**, arXiv:2105.01757. [CrossRef]
9. Gasparetto, A.; Scalera, L. A Brief History of Industrial Robotics in the 20th Century. *Adv. Hist. Stud.* **2019**, *8*, 24–35. [CrossRef]
10. Ballestar, M.T.; Díaz-Chao, A.; Sainz, J.; Torrent-Sellens, J. Impact of Robotics on Manufacturing: A Longitudinal Machine Learning Perspective. *Technol. Forecast. Soc. Chang.* **2020**, *162*, 120348. [CrossRef]
11. Pedersen, M.R.; Nalpantidis, L.; Andersen, R.S.; Schou, C.; Bøgh, S.; Krüger, V.; Madsen, O. Robot Skills for Manufacturing: From Concept to Industrial Deployment. *Robot. Comput. Integr. Manuf.* **2006**, *37*, 282–291. [CrossRef]
12. Mohammed, M.Q.; Chung, K.L.; Chyi, C.S. Review of Deep Reinforcement Learning-Based Object Grasping: Techniques, Open Challenges, and Recommendations. *IEEE Access* **2020**, *8*, 178450–178481. [CrossRef]
13. Liu, R.; Nageotte, F.; Zanne, P.; de Mathelin, M.; Dresp-Langley, B. Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review. *MDPI Robot.* **2021**, *10*, 1–13.
14. Tai, L.; Zhang, J.; Liu, M.; Boedecker, J.; Burgard, W. Survey of Deep Network Solutions for Learning Control in Robotics: From Reinforcement to Imitation. *arXiv* **2016**, arXiv:1612.07139.
15. Sutton, R.; Barto, A. *Reinforcement Learning: An Introduction*; The MIT Press: Cambridge, MA, USA; London, UK, 2018.
16. Kober, J.; Bagnell, A.; Peters, J. Reinforcement Learning in Robotics: A Survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
17. Fu, J.; Levine, S.; Abbeel, P. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. Proceeding of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; IEEE: New York, NY, USA, 2016; pp. 4019–4026.
18. Lewis, F.; Dawson, D.; Abdallah, C. *Robotic Manipulator Control Theory and Practice*, 2nd ed.; Revised and Expanded; Marcel Kekker, Inc.: New York, NY, USA, 2005.
19. Gualtieri, M.; Pas, A.; Platt, R. *Pick and Place without Geometric Object Models*; IEEE: Brisbane, QLD, Australia, 2018; pp. 7433–7440.
20. Stapelberg, B.; Malan, K.M. A Survey of Benchmarking Frameworks for Reinforcement Learning. *South Afr. Comput. J.* **2020**, *32*. [CrossRef]
21. Ng, A.Y.; Harada, D.; Russell, S. Policy Invariance under Reward Transformations Theory and Application to Reward Shaping. In Proceedings of the Sixteenth International Conference on Machine Learning, San Francisco, CA, USA, 27–30 June 1999; pp. 278–287.
22. Gualtieri, M.; Platt, R. Learning 6-DoF Grasping and Pick-Place Using Attention Focus. In Proceedings of the 2nd Conference on Robot Learning, Zürich, Switzerland, 29 October 2018.
23. Kleeberger, K.; Bormann, R.; Kraus, W.; Huber, M. A Survey on Learning-Based Robotic Grasping. *Curr. Robot. Rep.* **2020**, 239–249. [CrossRef]

24. Atkeson, C.; Santamaria, J. A Comparison of Direct and Model-Based Reinforcement Learning. In Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, NM, USA, 25 April 1997.
25. Sigaud, O.; Buffet, O. *Markov Decision Processes in Artificial Intelligence*, 2nd ed.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2010.
26. Russell, S.; Norvig, P. *Artificial Intelligence A Modern Approach*, 4th ed.; Pearson Education, Inc.: Hoboken, NJ, USA. ISBN 978-0-13-461099-3.
27. Deisenroth, M.P.; Neumann, G.; Peters, J. A Survey on Policy Search for Robotics. *Found. Trends Robot.* **2013**, *2*, 1–114. [CrossRef]
28. Levine, S.; Koltun, V. Guided policy search. In Proceedings of the Machine Learning Research, Journal of Machine Learning Research, Atlanta, GA, USA, 16 June 2013; Volume 28, pp. 1–9.
29. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1928–1937.
30. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 16 June 2016; Volume 32.
31. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Oleg Klimov Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
32. Laud, A.D. *Theory and Application of Reward Shaping in Reinforcement Learning*; University of Illinois at Urbana-Champaign: Champaign, IL, USA, 2004. ISBN 0496782142.
33. Nagpal, R.; Krishnan, A.U.; Yu, H. Reward Engineering for Object Pick and Place Training. *arXiv* **2020**, arXiv:2001.03792.
34. Grzes, M.; Kudenko, D. Learning shaping rewards in model-based reinforcement learning. In Proceedings of the AAMAS 2009 Workshop on Adaptive Learning Agents, Budapest, Hungary, 12 May 2009; Volume 115, p. 30.
35. Mataric, M.J. Reward functions for accelerated learning. In *Machine Learning Proceedings, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, 10–13 July 1994*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 181–189.
36. Luo, Y.; Dong, K.; Zhao, L.; Sun, Z.; Zhou, C.; Song, B. Balance between Efficient and Effective Learning: Dense2sparse Reward Shaping for Robot Manipulation with Environment Uncertainty. *arXiv* **2020**, arXiv:2003.02740.
37. Jang, S.; Han, M. Combining reward shaping and curriculum learning for training agents with high dimensional continuous action spaces. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, 17–19 October 2018; IEEE: New York, NY, USA, 2018; pp. 1391–1393.
38. Tenorio-Gonzalez, A.C.; Morales, E.F.; Villasenor-Pineda, L. Dynamic Reward Shaping: Training a Robot by Voice. In Proceedings of the Ibero-American Conference on Artificial Intelligence, Bahía Blanca, Argentina, 1–5 November 2010; Springer: New York, NY, USA, 2010; pp. 483–492.
39. Konidaris, G.; Barto, A. Autonomous shaping: Knowledge transfer in reinforcement learning. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 489–496.
40. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement Learning: A Survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
41. Ng, A.; Russell, S. Algorithms for Inverse Reinforcement Learning. In Proceedings of the Seventeenth International Conference on Machine Learning, San Francisco, CA, USA, 29 June–2 July 2000.
42. Osa, T.; Pajarinen, J.; Neumann, G.; Bagnell, J.A.; Abbeel, P.; Peters, J. An Algorithmic Perspective on Imitation Learning. *Found. Trends Robot.* **2018**, *7*, 1–179. [CrossRef]
43. Ho, J.; Ermon, S. Generative Adversarial Imitation Learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 4565–4573.
44. Stephane Ross Interactive Learning for Sequential Decisions and Predictions. Ph.D. Thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2013.
45. Pomerleau, D.A. *Alvinn: An Autonomous Land Vehicle in a Neural Network*; Technical Report; Carnegie—Mellon University, Artificial Intelligence and Psychology: Pittsburgh, PA, USA, 1989.
46. Farag, W.; Saleh, Z. Behavior Cloning for Autonomous Driving Using Convolutional Neural Networks. In Proceedings of the 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakhier, Bahrain, 18–19 November 2018; IEEE: New York, NY, USA, 2018.
47. Abbeel, P.; Ng, A.Y. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the Twenty-First International Conference on Machine Learning, Banff, AB, Canada, 4–8 July 2004; p. 1.
48. Bohg, J.; Morales, A.; Asfour, T.; Kragic, D. Data-Driven Grasp Synthesis—A Survey. *IEEE Trans. Robot.* **2016**, *30*, 289–309. [CrossRef]
49. Hodan, T.; Matas, J.; Obdrzalek, S. On evaluation of 6D object pose estimation. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Cham, Switzerland, 2016.
50. Brégier, R.; Devernay, F.; Leyrit, L.; Crowley, J.L. Defining the Pose of Any 3D Rigid Object and an Associated Distance. *Int. J. Comput. Vis.* **2017**, *126*, 571–596. [CrossRef]
51. Gualtieri, M.; Ten Pas, A.; Saenko, K.; Platt, R. High precision grasp pose detection in dense clutter. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; IEEE: New York, NY, USA, 2016.
52. Suarez, R.; Roa, M. Grasp Quality Measures: Review and Performance. *Auton. Robot.* **2014**, *38*, 65–88.
53. Sahbani, A.; El-Khoury, S.; Bidaud, P. An Overview of 3D Object Grasp Synthesis Algorithms. *Robot. Auton. Syst.* **2011**, *60*, 326–336. [CrossRef]

54. Jiang, Y.; Moseson, S.; Saxena, A. Efficient grasping from rgbd images: Learning using a New Rectangle Representation. In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011.
55. Zeng, A.; Song, S.; Yu, K.-T.; Donlon, E.; Hogan, F. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; IEEE: New York, NY, USA, 2018; pp. 3750–3757.
56. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; IEEE: New York, NY, USA, 2017; pp. 23–30.
57. Huang, S.-W.; Lin, C.-T.; Chen, S.-P.; Wu, Y.-Y.; Hsu, P.-H.; Lai, S.-H. Cross Domain Adaptation with GAN-Based Data Augmentation. In Proceedings of the Lecture Notes in Computer Science: Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; Springer: New York, NY, USA, 2018; Volume 11213, ISBN 978-3-030-01240-3.
58. Ivaldi, S.; Padois, V.; Nori, F. *Tools for Dynamics Simulation of Robots: A Survey based on User Feedback*; IEEE: Madrid, Spain, 2014; pp. 842–849.
59. Erez, T.; Tassa, Y.; Todorov, E. Simulation tools for model-based robotics: Comparison of bullet, Havok, MuJoCo, ODE and PhysX. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; IEEE: New York, NY, USA, 2015; pp. 4397–4404.
60. Popov, I.; Heess, N.; Lillicrap, T.; Hafner, R.; Barth-Maron, G.; Vecerik, M.; Lampe, T.; Tassa, Y.; Erez, T.; Riedmiller, M. Data-Efficient Deep Reinforcement Learning for Dexterous Manipulation. *arXiv* **2017**, arXiv:1704.03073.
61. Mahler, J.; Goldberg, K. Learning deep policies for robot bin picking by simulating robust grasping sequences. In Proceedings of the 1st Annual Conference on Robot Learning, Mountain View, CA, USA, 13 November 2017; pp. 515–524.
62. Sehgal, A.; La, H.; Louis, S.; Nguyen, H. Deep reinforcement learning using genetic algorithm for parameter optimization. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; IEEE: New York, NY, USA, 2019.
63. Zuo, G.; Lu, J.; Chen, K.; Yu, J.; Huang, X. Accomplishing robot grasping task rapidly via adversarial training. In Proceedings of the 2019 IEEE International Conference on Real-Time Computing and Robotics, Irkutsk, Russia, 4 August 2019.
64. Chen, C.; Li, H.Y.; Zhang, X.; Liu, X.; Tan, U.X. Towards robotic picking of targets with background distractors using deep reinforcement learning. In Proceedings of the 2nd WRC Symposium on Advanced Robotics and Automation 2019, Beijing, China, 21 August 2019; IEEE: New York, NY, USA, 2019.
65. Xiao, Y.; Katt, S.; Ten Pas, A.; Chen, S.; Amato, C. Online planning for target object search in clutter under partial observability. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20 May 2019; IEEE: New York, NY, USA, 2019.
66. Liu, D.; Wang, Z.; Lu, B.; Cong, M.; Yu, H.; Zou, Q. A Reinforcement Learning-Based Framework for Robot Manipulation Skill Acquisition. *IEEE Access* **2020**, *8*, 108429–108437. [CrossRef]
67. Mohammed, M.Q.; Chung, K.L.; Chyi, C.S. Pick and Place Objects in a Cluttered Scene Using Deep Reinforcement Learning. *Int. J. Mech. Mechatron. Eng.* **2020**, *20*, 50–57.
68. Li, B.; Lu, T.; Li, J.; Lu, N.; Cai, Y.; Wang, S. ACDER: Augmented curiosity-driven experience replay. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 4218–4224.
69. Pore, A.; Aragon-Camarasa, G. On simple reactive neural networks for behaviour-based reinforcement learning. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 August 2020.
70. Al-Selwi, H.F.; Aziz, A.A.; Abas, F.S.; Zyada, Z. Reinforcement learning for robotic applications with vision feedback. In Proceedings of the 2021 IEEE 17th International Colloquium on Signal Processing & Its Applications (CSPA), Langkawi, Malaysia, 5 March 2021; IEEE: New York, NY, USA, 2021.
71. Marzari, L.; Pore, A.; Dall'Alba, D.; Aragon-Camarasa, G.; Farinelli, A.; Fiorini, P. Towards Hierarchical Task Decomposition Using Deep Reinforcement Learning for Pick and Place Subtasks. *arXiv* **2021**, arXiv:2102.04022.
72. Anca, M.; Studley, M. Twin delayed hierarchical actor-critic. In Proceedings of the 2021 7th International Conference on Automation, Robotics and Applications (ICARA), Prague, Czech Republic, 4 February 2021; IEEE: New York, NY, USA, 2021.
73. Morrison, D.; Corke, P.; Leitner, J. Closing the Loop for Robotic Grasping: A Real-Time, Generative Grasp Synthesis Approach. *arXiv* **2018**, arXiv:1804.05172.
74. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Proceedings of the 2nd Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018.
75. Finn, C.; Levine, S.; Abbeel, P. Guided Cost Learning: Deep inverse optimal control via policy optimization. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; Volume 48. JMLR.
76. Wu, B.; Akinola, I.; Allen, P.K. Allen pixel-attentive policy gradient for multi-fingered grasping in cluttered scenes. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4 November 2019; IEEE: New York, NY, USA, 2019.
77. Deng, Y.; Guo, X.; Wei, Y.; Lu, K.; Fang, B.; Guo, D.; Liu, H.; Sun, F. Deep reinforcement learning for robotic pushing and picking in cluttered environment. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4 November 2019.

78. Beltrain-Hernandez, C.; Damien, P.; Harada, K.; Ramirez-Alpizar, I. Learning to Grasp with Primitive Shaped Object Policies. *2019 IEEE/SICE Int. Symp. Syst. Integr.* **2019**, 468–473. [CrossRef]
79. Berscheid, L.; Meißner, P.; Kröger, T. Robot learning of shifting objects for grasping in cluttered environments. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, 3–8 November 2019.
80. Kim, T.; Park, Y.; Park, Y.; Suh, I.H. Acceleration of Actor-Critic Deep Reinforcement Learning for Visual Grasping in Clutter by State Representation Learning Based on Disentanglement of a Raw Input Image. *arXiv* **2020**, arXiv:2002.11903v1.