A Survey of Action Spaces for Reinforcement Learning in Robotics

Schultze Jan
Forschungszentrum Informatik FZI
Karlsruhe, Germany
uctrj@student.kit.edu

***Abstract*—This survey investigates the design of continuous action spaces for reinforcement learning (RL) in robotics. Position- and Torque-based action spaces (defined either in joint or Cartesian space) are the predominant choice in the literature. Alternative formulations such as velocity-based action spaces as well as action spaces based on a variable impedance controller have received comparatively little attention.**

Recent work suggests that, although many action spaces are comparable in expressive power, they can differ substantially in learning speed, sample efficiency and sim-to-real transferability, particularly in contact-rich tasks.

This paper provides a structured overview of existing continuous action space designs and proposes a taxonomy based on representation, controlled quantity, parameterization and temporal abstraction. The survey does not propose a new action space, but aims to organize existing design choices and extract actionable insights for practitioners selecting action spaces for robotic reinforcement learning.

***Index Terms*—Reinforcement Learning, Robotics, Action Space**

I. Introduction

For decades, robot control has predominantly relied on model-based approaches, where motor commands are derived from explicit system models and predefined control laws. With the rapid development of artificial intelligence, reinforcement learning (RL) has gained increasing attention as a data-driven alternative for control policy learning. Instead of formulating an underlying system model, the control policy is learned directly from interaction data of an agent with its environment. RL is commonly formulated as a Markov Decision Process (MDP), in which the agent and its environment are described by a state representation that satisfies the Markov property. The Markov property states that the future state of the system depends solely on the current state and the action taken, making the history of past states and actions irrelevant for predicting the next transition. Mathematically, this is expressed as:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = P(s_{t+1}|s_t, a_t)$$

An agent (robot) in this environment can (partially) observe its current state and selects between a set of actions to change its state in order to increase its reward.

In the context of reinforcement learning these observations and states differ depending on the morphology of a robot. For example a wheel-based robot and a legged robot are fundamentally different in their state representation and therefore in their state space.

Recent research has primarily focused on improving the observation space [1, 2], by improving sensor data, evolving computer vision models and reducing noise. The underlying assumption is that if the robot understands its own state better, it will be capable of learning better policies and making better choices.

However, comparatively little attention has been paid to the design of the action space, even though the action space design influences system dynamics, stability and sim-to-real transfer [3]. When developing a policy for a robot, the action space is usually chosen by intuition. For instance, a wheeled robot might be associated with a wheel velocity action space, a legged robot with a joint position action space, and a manipulator with targets in a Cartesian position action space [4]. The selection is based on prior knowledge and existing controllers rather than on systematic evaluation criteria.

The most common action spaces for learning an optimal policy in RL are the joint position space and joint torque space. This raises the question whether policy performance can be systematically improved through principled action space design.

In this paper, we focus only on continuous action spaces that directly control the robot's actuators. Discrete action spaces have been excluded and high-level approaches like Dynamical Movement Primitives (DMPs) and latent action spaces are only featured for their classification. This survey does not propose a new action space, but aims to structure existing design choices (by introducing a taxonomy) and extract actionable insights for practitioners.

II. Background

A. Markov Decision Process

A Markov Decision Process (MDP) provides the framework for sequential decision making and is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \rho_0)$.

Here, $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ the set of possible actions, $P(s_{t+1} \mid s_t, a_t)$ the probability of reaching state s_{t+1} from s_t after action a_t, and $R(s_t, a_t)$ the reward received for that action. The discount factor $\gamma \in [0, 1]$ balances the importance of immediate versus future rewards: low values make the agent short-sighted, high values favor long-term

planning. Finally ρ_0 is the initial state distribution, specifying how likely the agent is to start in each state, i.e., $s_0 \sim \rho_0$ [5].

B. The Concept of Reinforcement Learning

In Reinforcement Learning (RL), an agent interacts with an environment modeled as a stochastic MDP. In contrast to a deterministic MDP, this means that applying the same action in the same state can lead to different next states, due to inherent uncertainties, sensor noise, partial observability, or environmental variations.

The transition dynamics P are unknown and must be learned through experience, meaning the agent cannot predict the outcome of its actions in advance. The reward function R is given.

At each discrete time step t, the agent observes the current state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to its policy $\pi(a \mid s)$. The environment then transitions to the next state s_{t+1}. After the transition, the agent receives a scalar reward $r_t = R(s_t, a_t)$.

Experience is collected in the form of (s_t, a_t, r_t, s_{t+1}) samples. The transition follows the environment dynamics $P(s_{t+1} \mid s_t, a_t)$.

1) Objective of RL: The goal of RL is to learn an optimal policy $\pi^*(a \mid s)$ that specifies which actions to take in each state. A policy is optimal when it maximizes the expected sum of discounted long-term rewards over initial states drawn from ρ_0 [6]:

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0}\left[V^{\pi}(s_0)\right] = \mathbb{E}_{s_0 \sim \rho_0}\left[\mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0\right]\right].$$

2) Learning Methods: Several algorithms have been proposed for learning good or even optimal policies from (s_t, a_t, r_t, s_{t+1}) samples. In *continuous* control problems, *gradient methods* are commonly used. A widely applied algorithm is *Proximal Policy Optimization (PPO)*, which stabilizes training by constraining the magnitude of policy updates between iterations, preventing destructive updates [7].

C. State Space, Observation Space, Action Space

"What is a state space?" The state space is the set of all possible configurations of the environment that fully describe its current situation. For example, in a robotic arm, a state could include all joint angles and velocities.

"What is an observation space?" The observation space is the set of all information the agent can perceive from the environment, which may only partially represent the true state. This means that the observation space is typically a partial or transformed representation of the state space. For example, a robot may only observe the end-effector position through a camera, rather than the full joint configuration. In this case, the camera acts as a transformation from the state space to the observation space.

"What is an action space?" An action space defines all the possible actions an agent can take in an environment. It can be discrete (a finite set of actions, like "move left/right") or continuous (a range of values, like joint angles or velocities). For robotics the action space is usually continuous. For example a manipulator has a range of joint angles, end effector velocities and joint torques it can reach. The choice of action space is made by the engineer who develops the policy and shapes how the agent interacts with the environment and affects learning efficiency and policy performance.

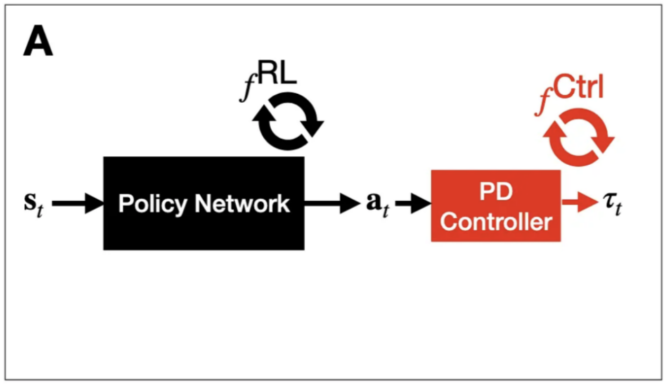

Fig. 1. Action Space Illustrative Diagram depicts the hierarchical architecture of the policy network and control feedback loop [4]. The RL policy outputs actions, which are processed through a feedback control loop before being applied to the robot's actuators.

D. Control Architecture and Abstraction Levels

Robotic systems typically employ a hierarchical control architecture where the RL policy and low-level control feedback loops operate at different abstraction levels and time scales.

The **RL policy** operates at a relatively low frequency (e.g., 10–100 Hz) and generates high-level commands that specify the desired goal or reference state. These outputs may take various forms: desired end-effector poses x_{des}, desired joint positions q_{des}, desired joint velocities $\dot{q}_{des}$, or impedance parameters (controller gains) that specify the intended interaction behavior.

A **feedback control loop** sits between the policy output and the robot's actuators, operating at a significantly higher frequency (e.g., 500–2000 Hz). This control layer tracks the reference signals from the policy and converts them into physical actuation commands, typically joint torques τ. Common implementations include proportional-derivative (PD) control in joint space, impedance control in task space, and inverse dynamics control.

Different action space designs correspond to different positions in this hierarchy. Joint-torque-based action spaces place the policy at the lowest level, directly commanding torques with minimal feedback control. In contrast, position-, velocity-, or impedance-based action spaces shift the policy to a higher abstraction level, with more sophisticated feedback loops handling the dynamics and control details.

The gains of feedback controllers can be fixed (predetermined) or variable (learned as part of the policy). Variable gains are particularly beneficial for contact-rich tasks, allowing

the controller to adapt its behavior online in response to changing contact conditions.

III. Taxonomy of Action Spaces

Figure 2 shows the taxonomy of action spaces and is used as the basis for the following explanation.

Definition of Terminology: Throughout this survey, we distinguish between two fundamental classes of action spaces:

- **Direct Control Action Spaces:** The policy directly outputs the controlled physical quantity (position, velocity, or force/torque) that should be applied or achieved by the robot at each control step. These action spaces operate at the lowest level of temporal abstraction and are applied directly to the actuators (either immediately or through a simple feedback control loop such as PD control).
- **Abstract Action Spaces:** The policy outputs higher-level goals, trajectory parameters, or latent representations that must be decoded or processed into direct control commands. Examples include Dynamic Movement Primitives (DMPs), which output trajectory parameters that are integrated over time, and latent action spaces, which use learned decoders (e.g., variational autoencoders) to map latent codes into control signals.

This distinction is orthogonal to the presence of feedback control loops. Both direct and abstract action spaces may be combined with various feedback control mechanisms (PD, impedance, inverse dynamics, etc.). The terminology focuses on the information content and abstraction level of the policy's output, not on the control architecture that follows.

A. Action Domain

The action domain distinguishes whether the policy outputs discrete actions or continuous actions. The difference is that **discrete actions are countable**. E.g. policy outputs could be the choice of control modes, gear switching or digital outputs [8]. For example in a simple game called MountainCar-V0 environment of Atari, there are three actions: push the car to the left, push the car to the right, and stand still. In the **continuous action** space problem, we **cannot count the number of actions, but we can describe their range**. This can be illustrated briefly by a manipulation task of a robotic arm. The output action of a robot arm includes angles from 0 to 360 degrees and force from 0 to a certain strength [9].

While most robotic control problems rely on continuous action spaces, discrete or hybrid domains are commonly used for high-level decision making and hierarchical reinforcement learning.

B. Temporal Abstraction

In this survey we focus on direct control action spaces, meaning the policy directly controls the physical quantities: position, velocity, or force/torques. More abstract approaches using DMPs (Dynamic Movement Primitives) or latent action spaces are included for classification purposes but not analyzed in detail.

C. Space of Representation

1) Cartesian Space / Task Space / End-Effector Space: In task space, actions are defined in cartesian coordinates (x, y, z) and specify the desired motion or interaction of the robot's end-effector. Examples include desired positions, velocities, or forces. Task-space actions are intuitive for manipulation because they are represented in 3D coordinates (related to sensor representations, such as from a depth camera), but require inverse kinematics or inverse dynamics to be mapped to joint torques.

2) Joint / Configuration Space: In configuration space, actions are defined directly at the joint level and control individual joints. Typical representations include desired joint positions q, velocities $\dot{q}$, or torques τ. In some systems, actions are defined at the actuation level rather than at the joint level. A notable example is muscle activation spaces, where actions specify activation signals $a_{\text{MTU}} \in [0, 1]$ for musculotendon units. Joint torques arise implicitly from the nonlinear muscle dynamics, introducing redundancy and compliance but increasing learning complexity [10]. Joint space provides direct access to the robot's actuated degrees of freedom. The complex dynamics of the robot are not addressed explicitly through an inverse dynamics controller but instead are learned implicitly by the policy.

D. Controlled Quantity

Direct control action spaces can be divided into cartesian-based and joint-based representations. For both coordinate systems, a different physical state can be chosen as the controlled quantity: position, velocity, or force/torque. This results in six basic direct control action spaces. Those can be further divided into absolute and delta action spaces. Delta action spaces are explained in more detail in the Action Parameterization subsection below.

E. Action Parameterization

Action parameterization defines the mapping from policy output to physical control commands. While **absolute control** specifies fixed target values (e.g., a 90° joint angle), **delta control** commands a relative change (e.g., +2°). Following the taxonomy of Aljalbout et al. [5] these delta action spaces are categorized by their integration reference into *One-step Integrators* (OIΔ) and *Multi-step Integrators* (MIΔ).

Both variants compute a target vector $\vec{v}_d$ (e.g., $q_d, \dot{q}_d, x_d, \dot{x}_d$) using the policy output $a \in [-1, 1]$, a step duration dt, and a scaling hyperparameter c to ensure smoothness. The distinction lies in the source of the reference vector $\vec{v}$:

- **One-step Integrator (OIΔ):** Relies on the **current system feedback** $\vec{v}_{meas}$ where the next command is always relative to the robot's actual physical state.

$$\vec{v}_d \leftarrow \vec{v}_{meas} + c \cdot a \cdot dt \tag{1}$$

- **Multi-step Integrator (MIΔ):** Relies on the **previous policy target** where targets are integrated recurrently,

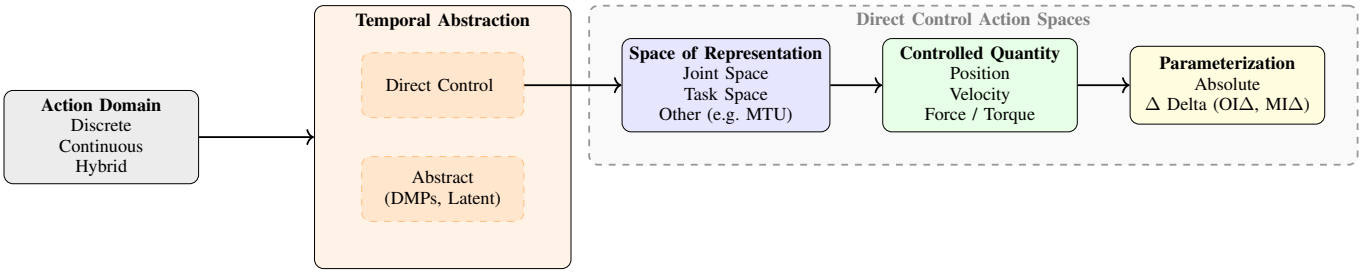

Fig. 2. Taxonomy of action spaces. The temporal abstraction layer defines whether the policy outputs direct control commands (position, velocity, or force at each timestep) or abstract representations (DMPs, latent codes) that are decoded into control signals. Direct control action spaces are further specified by their space of representation, controlled quantity, and parameterization.

allowing the desired trajectory to evolve independently of the robot's actual pose.

$$\vec{v}_{d,t+1} \leftarrow \vec{v}_{d,t} + c \cdot a \cdot dt \tag{2}$$

IV. Comparison of Action Spaces for Robotics

A. Direct Control Action Spaces

Eser et al. [4] found different action spaces to be comparable in their expressiveness but they can differ in their learning dynamics. For practitioners, this means: that one action space might enable faster learning of a task but another action space might be better suited for stable execution. **Both action spaces will eventually converge to similar policies. But the quicker learned policy can be distilled in another action space using imitation learning saving computational resources**. Special about their work is, that they introduce a Generalized Parameterization of Action Spaces. This is the mathematical base of the translation of action spaces.

Aljabout et al. [3] found that learning only trajectories does not work well for manipulation tasks; these tasks require information about contact forces, not just position. For contact-rich tasks, position-based action spaces are unsuitable.

In a related study, Aljabout et al. [5] compared 13 out of 18 possible action space configurations. Training of manipulation tasks was conducted in simulation, and sim-to-real transferability was evaluated. **They found Cartesian Velocity to perform best in simulation and Joint Velocity to be the best-performing action space in sim-to-real transfer**.

Peng et al. [10] compared action spaces on locomotion tasks (humanoid, raptor, dog). Notable is musculotendon units (MTU), which, due to their complexity, learned significantly slower. Complementary to Varin et al., **direct torque control performed worst**, often failing to reach comparable success rates. **PD control performed best**, with highest sample efficiency and best scaling to complex morphologies. The findings align with those of Eßer et al. [4]: different action spaces mainly vary in learning speed but reach the same expressiveness.

Feng et al. [11] conducted a large-scale empirical study on imitation learning and direct control action spaces. A key finding is that **delta action spaces perform consistently better than absolute action spaces** when implemented correctly, with chunk-wise delta superior to step-wise delta. They conclude that **joint space representation with chunk-wise delta parameterization** achieves the most robust results.

B. Hierarchical Control with Feedback Loops

Martín-Martín et al. [1] compare different action spaces with a focus on contact rich tasks (door opening, surface wiping). They strongly advocate for variable impedance control action spaces combined with feedback control loops. They make a detailed comparison between different fixed gain and variable gain impedance control structures and even consider the space of representation in their experiments. The performance in both joint- and task-space is similar but the sim to real transfer works better for task-space. That is why **they conclude that (variable) impedance policies learned in cartesian space are independent of the dynamic effects of the robot and therefore independent of its morphology and best suited for contact rich tasks**.

Ulmer et al. [12] **introduced a bio-inspired action space with a hierarchical control architecture for contact-rich manipulation tasks**. The adaptive force impedance controller in cartesian space was tested in real life on a wiping task and in simulation on a door opening task, object lifting task and again the wiping task. The key idea of this paper is to introduce a hierarchical policy that is capable of handling multimodal signals to reduce complexity. The policy is factored into two components. A slow outer loop which plans the correct actions to solve a given task. And a fast inner loop which continuously adapts the interaction to minimize the trajectory error (and consumed energy).

Varin et al. [13] found **the impedance controller action space to have learned the fastest and a classic torque controller learning three times slower than the impedance controller when trained with PPO**. A general finding from their work is that PD control is a better action space than direct torque control for locomotion tasks. The reason why the impedance controller learns so fast is due to the fact that it is defined in cartesian space, same as the reward term. The straightforward mapping between rewards and actions leads to a faster exploration phase.

Bogdanovic et al. [2] studied action space selection for contact-rich tasks, focusing on uncertainties in contact interactions (friction, stiffness and contact location). The policy must solve two tasks: a manipulation task where a robot wipes a table in simulation, and a locomotion task where a single-legged robot repeatedly jumps to a target height. For the jumping task, online PD gain adaptation is crucial because the ground height varies between jumps, introducing contact uncertainties. They compared a direct torque controller with fixed and variable impedance control. **While, in theory, the direct torque controller could learn both tasks, it often fails in practice**. For the wiping task, both fixed and variable impedance controllers performed similarly, as there were no uncertainties. However, **in the presence of uncertainties, the variable impedance controller clearly outperformed the others, demonstrating its robustness for tasks like the jumping scenario**. Additionally, they introduced an auxiliary reward term that encourages smooth force profiles and allows for small overshoots during contact establishment and breaking.

C. Abstract Action Spaces

In a related study, Allshire et al. [14] introduce a method to learn latent action spaces for efficient reinforcement learning called LASER. The core principle is the transformation of the original MDP of an RL problem into a new MDP, where exploration is easier. The variational encoder-decoder model maps actions into a latent action space. Policy search is then performed in this latent action representation. The model is trained in two variants: *offline* and *online*. In the online variant, LASER is trained simultaneously with the policy. In the offline variant, LASER learns from a dataset of expert policies and is then able to transfer this knowledge to new tasks (door opening and surface wiping). This approach has the advantage that **LASER enables faster convergence and improved exploration for transfer learning tasks**.

Aljabout et al. [15] introduce CLAS: a centralized latent action space for multi robot manipulation. The main benefit of their approach is the increased sample efficiency and improved coordination in high-dimensional tasks. Unlike traditional multi-agent reinforcement learning (MARL), which often struggles with decentralized action generation, CLAS learns a latent representation of the task which is independent of the robot's morphology. By using a conditional variational autoencoder (CVAE), the framework compresses the joint action space of multiple robots into a lower-dimensional manifold that captures the core requirements of the manipulation task, such as the resulting wrench (torque in Task Space) on a shared object.

During training, a centralized encoder maps the full state and all individual robot actions to this latent space. For execution, the system employs a centralized training, decentralized execution (CTDE) paradigm: while the policies operate in the shared latent space to ensure coordination, agent-specific decoders translate these latent commands back into the original action space of each robot.

This abstraction allows CLAS to outperform both naive single-agent approaches, which suffer from the "curse of dimensionality," and standard decentralized MARL baselines, which fail to achieve the necessary precision for physical interaction. Their results demonstrate that **CLAS is particularly effective in scaling to complex scenarios, such as four-arm manipulation, where it maintains stability and robustness against external disturbances that cause conventional methods to fail**.

D. Movement Primitives

Bahl et al. [16] introduce **Neural Dynamic Policies (NDPs)**, which embed the structure of classical dynamical systems directly into deep neural networks. Instead of predicting raw control signals at every timestep, the policy outputs the parameters of a **Dynamic Movement Primitive (DMP)**—specifically the goal (g) and the weights (w) of a forcing function. The key innovation is the end-to-end differentiability: an integrated Euler solver allows gradients to flow through the trajectory equations back to the neural network. This approach significantly improves **sample efficiency** and ensures smooth, temporally consistent motion.

E. Discretization of Continuous Action Spaces

Tang et al. [17] found that **discretizing continuous action spaces greatly improved the performance** compared to continuous / gaussian action spaces. This was especially noticeable on high dimensional tasks with complex dynamics (like grasping).

Zhu et al. [9] compare different RL algorithms according to their Action Space suitability. The RL algorithms are divided in three action domains: discrete, continuous and discrete-continuous hybrid. Algorithms for **discrete action spaces are mostly deep Q-learning algorithms (DQN)**. Algorithms for **continuous action spaces are based on policy gradient methods**. For hybrid action space, researchers generally modify the previous algorithms in order to output appropriate actions.

Neunert et al. [8] researched hybrid RL, demonstrating that treating hybrid problems in their native form outperforms artificial continuization. A key finding is that **the continuization of discrete action spaces can harm policy performance**. This is exemplified by the *gripper problem*: when a binary "open/close" gripper is modeled as a continuous range $[-1, 1]$, the policy's initial exploration (centered around 0) results in a hovering state where the gripper never fully opens or closes. By treating the gripper as a natively discrete action, the agent explores the extreme states directly, which is essential for discovering the sparse rewards of successful grasping.

Tang et al. [17] discretize the continuous range of action into a finite set of atomic actions and reduce the original task into a new task with a discrete action space. This is done for on-policy-optimization. They focus on the PPO/TRPO and ACKTR algorithms. They conclude that for tasks with relatively simple dynamics the discretization of the policy

does not necessarily improve the performance. But **for high-dimensional tasks with very complex dynamics, discrete policy significantly outperforms Gaussian policy**.

V. Comparison of Action Spaces outside of robotics

Dulac-Arnold et al. [18] researched large discrete action spaces and **found a way to transform discrete into continuous action spaces using prior information**. Then nearest neighbour search is performed to select the best action.

Karnevisto et al. [19] researched action space shaping in video games. This means they removed actions, combined multiple actions into one action and they discretized continuous actions. The reason for that is best explained by the following analogy: "Reducing the number of buttons [the agent can press] might ease the learning but comes with the risk of limiting the agent's performance". **Their most important finding was that continuous action spaces are harder to learn than discretized action spaces. And discretizing a continuous action space can notably increase performance**.

VI. Discussion

The action space influences learning dynamics and policy performance. In general, combining direct control action spaces with hierarchical feedback control loops (PD, impedance, or inverse dynamics) makes practical sense: the policy decides what commands to output while the feedback controller handles dynamics, reducing learning complexity.

An impedance controller combined with task-space action spaces also helps during the exploration phase. Policies learned in task/cartesian space only need to learn to solve the task problem. Additionally, they perform relatively well in sim-to-real transfer since the policies are independent of the robot's embodiment. In contrast, policies learned in joint space must additionally learn the inverse kinematics problem. This explains the recent trend towards variable impedance controllers (in task space) in robotics.

An additional benefit of feedback control loops is that they produce smooth, more natural motions, which are often not explicitly captured in the reward term.

For contact-rich tasks, it is helpful to consider variable gains for the controllers. Here the gains are learned as part of the policy. The variability of the gains is crucial during contact establishment.

Theoretically, all direct control action spaces are similar in expressiveness and converge to similar policies. In practice, they mainly differ in their learning dynamics. However, direct torque control often exhibits aggressive or jittery movements in practice, which is safety-critical for sim-to-real transfer. Sometimes direct torque control is even unable to converge to a sensible policy at all.

Due to practical challenges with direct torque control (such as jittery movements and numerical instability), a clear trend can be identified away from direct control action spaces towards higher-level abstractions like abstract action spaces (latent spaces, DMPs) that are embedded in modern neural network models, such as variational autoencoders with encoder-decoder structures.

VII. Practical Guidelines

Choice of Action Space: Direct control action spaces generally offer similar expressivity and converge to comparable policies. However, **direct torque control** often exhibits high-frequency oscillations (chattering) and numerical instability, making it less robust than position or velocity-based action spaces.

Contact-Rich vs. Simple Tasks: Variable Impedance Control in task space is superior for **contact-rich tasks** by enhancing safety during contact establishment, although it increases computational complexity. For **simple locomotion**, fixed-gain controllers are preferable, providing smoother trajectories without the overhead of variable stiffness.

Policy Distillation: To balance learning speed and hardware stability, a policy can be rapidly trained in a sample-efficient action space (e.g., joint positions) and subsequently **distilled** into a more robust action space (e.g., impedance control) via imitation learning.

VIII. Limitations and Future Work

This survey highlights the diverse landscape of action spaces in robotic learning. However, current literature lacks standardized benchmarks: individual studies focus on specific action space subsets rather than comprehensive comparisons. Additionally, potential reporting bias exists, as papers introducing novel action spaces may prioritize favorable scenarios. Large-scale, independent benchmarking studies—such as those by Feng et al. [11]—are essential to establish consensus in the field.

Acknowledgment

This paper was supervised by Sabine Bellmann from FZI.

References

[1] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable Impedance Control in End-Effector Space: An Action Space for Reinforcement Learning in Contact-Rich Tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 1010–1017. [Online]. Available: https://ieeexplore.ieee.org/document/8968201/

[2] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning Variable Impedance Control for Contact Sensitive Tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6129–6136, Oct. 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9146673

[3] E. Aljalbout, F. Frank, P. v. d. Smagt, and A. Paraschos, "The Shortcomings of Force-from-Motion in Robot Learning," Jul. 2024, arXiv:2407.02904 [cs]. [Online]. Available: http://arxiv.org/abs/2407.02904

[4] J. Eßer, G. B. Margolis, O. Urbann, S. Kerner, and P. Agrawal, "Action Space Design in Reinforcement Learning for Robot Motor Skills," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=GGuNkjQSrk

[5] E. Aljalbout, F. Frank, M. Karl, and P. v. d. Smagt, "On the Role of the Action Space in Robot Manipulation Learning and Sim-to-Real Transfer," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5895–5902, Jun. 2024, arXiv:2312.03673 [cs]. [Online]. Available: http://arxiv.org/abs/2312.03673

[6] J. Pazis and M. G. Lagoudakis, "Reinforcement learning in multidimensional continuous action spaces," in *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, Apr. 2011, pp. 97–104. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5967381

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, arXiv:1707.06347 [cs]. [Online]. Available: http://arxiv.org/abs/1707.06347

[8] M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller, "Continuous-Discrete Reinforcement Learning for Hybrid Control in Robotics," in *Proceedings of the Conference on Robot Learning*. PMLR, May 2020, pp. 735–751. [Online]. Available: https://proceedings.mlr.press/v100/neunert20a.html

[9] J. Zhu, F. Wu, and J. Zhao, "An Overview of the Action Space for Deep Reinforcement Learning," in *Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, ser. ACAI '21. New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3508546.3508598

[10] X. B. Peng and M. van de Panne, "Learning locomotion skills using DeepRL: does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, ser. SCA '17. New York, NY, USA: Association for Computing Machinery, Jul. 2017, pp. 1–13. [Online]. Available: https://doi.org/10.1145/3099564.3099567

[11] Y. Feng, J. Zheng, Z. Wang, D. Liu, J. Li, J. Pang, T. Wang, and X. Zhan, "Demystifying Action Space Design for Robotic Manipulation Policies," Feb. 2026, arXiv:2602.23408 [cs]. [Online]. Available: http://arxiv.org/abs/2602.23408

[12] M. Ulmer, E. Aljalbout, S. Schwarz, and S. Haddadin, "Learning Robotic Manipulation Skills Using an Adaptive Force-Impedance Action Space," Oct. 2021, arXiv:2110.09904 [cs]. [Online]. Available: http://arxiv.org/abs/2110.09904

[13] P. Varin, L. Grossman, and S. Kuindersma, "A Comparison of Action Spaces for Learning Manipulation Tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 6015–6021. [Online]. Available: https://ieeexplore.ieee.org/document/8967946/

[14] A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, and A. Garg, "LASER: Learning a Latent Action Space for Efficient Reinforcement Learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 6650–6656. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9561232

[15] E. Aljalbout, M. Karl, and P. v. d. Smagt, "CLAS: Coordinating Multi-Robot Manipulation with Central Latent Action Spaces," in *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*. PMLR, Jun. 2023, pp. 1152–1166. [Online]. Available: https://proceedings.mlr.press/v211/aljalbout23a.html

[16] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural Dynamic Policies for End-to-End Sensorimotor Learning," Dec. 2020, arXiv:2012.02788 [cs]. [Online]. Available: http://arxiv.org/abs/2012.02788

[17] Y. Tang and S. Agrawal, "Discretizing Continuous Action Space for On-Policy Optimization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5981–5988, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6059

[18] G. Dulac-Arnold, R. Evans, H. v. Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep Reinforcement Learning in Large Discrete Action Spaces," Apr. 2016, arXiv:1512.07679 [cs]. [Online]. Available: http://arxiv.org/abs/1512.07679

[19] A. Kanervisto, C. Scheller, and V. Hautamäki, "Action Space Shaping in Deep Reinforcement Learning," in *2020 IEEE Conference on Games (CoG)*, Aug. 2020, pp. 479–486. [Online]. Available: https://ieeexplore.ieee.org/document/9231687/