Learning Skills From Demonstrations: A Trend From Motion Primitives to Experience Abstraction

Mehrdad Tavassoli, Sunny Katyara, *Member, IEEE*, Maria Pozzi, *Member, IEEE*, Nikhil Deshpande, *Member, IEEE*, Darwin G. Caldwell, *Fellow, IEEE*, and Domenico Prattichizzo, *Fellow, IEEE*

***Abstract*—The uses of robots are changing from static environments in factories to encompass novel concepts such as human–robot collaboration in unstructured settings. Preprogramming all the functionalities for robots becomes impractical, and hence, robots need to learn how to react to new events autonomously, just like humans. However, humans, unlike machines, are naturally skilled in responding to unexpected circumstances based on either experiences or observations. Hence, embedding such anthropoid behaviors into robots entails the development of neuro-cognitive models that emulate motor skills under a robot learning paradigm. Effective encoding of these skills is bound to the proper choice of tools and techniques. This survey paper studies different motion and behavior learning methods ranging from movement primitives (MPs) to experience abstraction (EA), applied to different robotic tasks. These methods are scrutinized and then experimentally benchmarked by reconstructing a standard pick-n-place task. Apart from providing a standard guideline for the selection of strategies and algorithms, this article aims to draw a perspective on their possible extensions and improvements.**

***Index Terms*—Computational complexity, imitation learning, reinforcement learning.**

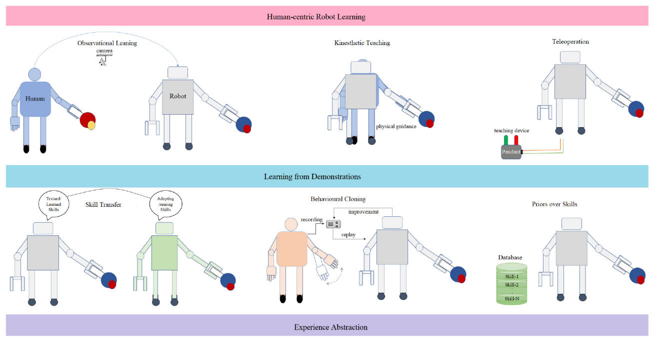


Fig. 1. Different paradigms of human-centric robot learning. Observational learning—humans show how to perform a task, and robot vision infers it, kinesthetic teaching—humans physically guide the robot toward task execution, teleoperation—the robot is actuated using its teach pendant for task demonstration, skill transfer—a robot shares its experience with another robot for generic task realization, BC—humans initialize the robot's policy with related actions of the task, and priors over skills—robots learn basic primitive skills from offline data for generalized task.

I. Introduction

OVER the past decade, the nature of tasks assigned to robots has changed, as has their workspace. Dynamic elements in their environment, like humans, objects, and other pieces of machinery, introduce various uncertainties to programming. Hence, to some extent, robots' autonomy would be forfeited, obliging them to learn how to react to events using experiences or skills that have been transferred to them.

Traditionally, robots preprogram the necessary skills to accomplish their tasks. Preplanning the robot's behavior might overfit its control to the given task, which reduces its self-sufficiency. Moreover, robots' workspaces have abundant dynamics, making it infeasible to hardcode every possible scenario. Instead, robots should learn the requisite skills to exchange with their surrounding elements. In this regard, learning algorithms evaluate every possible solution within the robot's reach. In such assessments, computational complexity exponentially grows for every extra Degree of Freedom (DoF), besides the time it takes to train. It is the dilemma that mainly affects robots with multiple DoFs like humanoids (e.g., WalkMan with 33, ASIMO with 34, and TALOS with 34 [1]).

Hence, robots need efficient learning algorithms that reduce the required time and memory by shrinking the learning problem to a subset of the whole solution space. In this regard, Demonstrating the task through mediums like kinesthetics, observations, teleoperation, or multimodal interface. After that, the system may incrementally learn complicated movements based on prior knowledge (Bayesian Inference) [2]. Although shrinking the state–action search space nurtures training quantitively, optimal results are contingent on choosing the proper initial conditions [3], as conceptually depicted in Fig. 1.

This article highlights the two main doctrines of learning skills from demonstrations: 1) movement primitives (MPs) and 2) experience abstraction (EA). Either method discards the irrelevant solution in the learning algorithms' search space by

Manuscript received 20 October 2022; revised 21 March 2023 and 27 May 2023; accepted 13 July 2023. Date of publication 18 July 2023; date of current version 12 February 2024. This work was supported in part by the Progetto Prin 2017 "TIGHT: Tactile Integration for Humans and Artificial Systems" under Grant 2017SB48FP. *(Corresponding author: Domenico Prattichizzo.)*

Mehrdad Tavassoli, Nikhil Deshpande, Darwin G. Caldwell, and Domenico Prattichizzo are with the Department of Advanced Robotics, Italian Institute of Technology, 16163 Genova, Italy (e-mail: mehrdad.tavassoli@iit.it; nikhil.deshpande@iit.it; darwin.caldwell@iit.it; domenico.prattichizzo@iit.it).

Sunny Katyara is with the Robotics and Automation Team, Irish Manufacturing Research, Dublin D24, D24 WCO4 Ireland (e-mail: sunny.katyara@imr.ie).

Maria Pozzi is with the Department of Information Engineering and Mathematics, University of Siena, 53100 Siena, Italy (e-mail: maria.pozzi@unisi.it).

Digital Object Identifier 10.1109/TCDS.2023.3296166

the given task demonstration. However, while MP focuses on learning the motion or task, EA learns sequences of primitives and skills that motivate the motion. In summary, EA lies one step ahead of MP in the learning control architecture of complex (loco-)manipulation problem.

Generally, learning from demonstration (LfD) can either encode the trajectory (low-level representation) or abstract the task by an array of subgoals known as key points (high-level representation) [4]. High-level representations are usually favored due to their generality. However, they fail to scale the learning algorithm well to systems with multiple DoFs [1]. High-level representations were first introduced under "symbolic reasoning" terminology [5], which encapsulates many aspects of the task, including task sequences and their underlying hierarchy [4]. The proper action is chosen at each stage (state), leading to discrete state–action–state transitions [1]. Each state then turns into an "if-then" statement to include the action's variability [4]. Nonetheless, for a general outcome, the interstate transition should be continuous [6], which leads to the concept of MPs.

MPs are simple reusable units of movement where each unit can be the building block of a more complex movement. MP holds a complete state–action representation through a limited number of parameters, assisting the learning process [7]. Hence, MP forms a modular control architecture. In many MP realizations, the learning algorithm scales up or down a specific MP block (predefined basic movement) to the segments of complex tasks. The composition of these blocks (in series or parallel) renders a complex task reproduction in new circumstances. MP methods can be categorized in multiple ways. Although, based on their tasks realization, MP approaches are divided into two main classes in [8]: 1) state-based MP [9], [10] and 2) trajectory-based MP [11], [12], [13].Trajectory-based MPs are best suited for tasks that require precise control and smooth motion generation, such as reaching or trajectory tracking. On the other hand, state-based MPs excel in tasks that involve frequent variations in the environment and require adaptability, such as obstacle avoidance or reactive manipulation.

MP approaches can be seen from other perspectives, like the number of parameters and their applications. Some MP methods employ a finite number of parameters in their formulation, e.g., Gaussian mixture model (GMM), whereas others may use infinite parameters, e.g., hidden Markov models (HMMs). In parametric approaches, the number of parameters is fixed regardless of the input size, but it increases when input expands in nonparametric ones [14]. Although their training data sets are bigger, parametric approaches offer better flexibility. Based on their application, MP methods either deduce the human's intention to act accordingly (intention learning) or emulate the observed behaviors in different plots (imitation learning). The former is mainly utilized in human–robot collaboration (HRC), and the latter is used where the robot should have some degree of mission autonomy.

In contrast to MP, EA leverages human behaviors to extract the skills and rewards for goal accomplishment. Different training strategies deliver EA algorithms with the

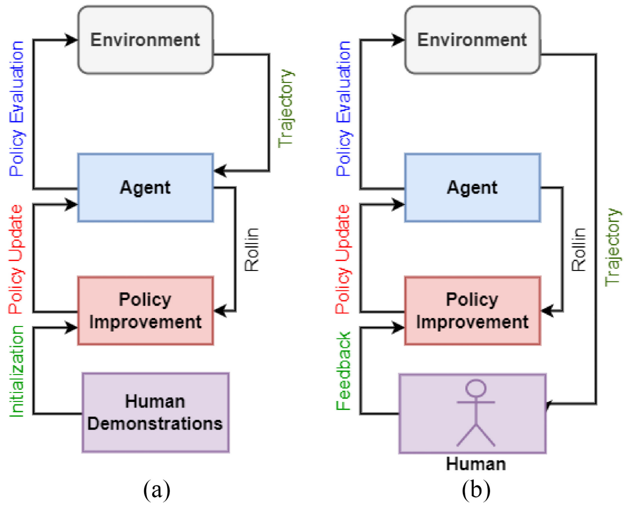


Fig. 2. Block diagram representation of EA in imitation learning. (a) Offline cogitation of human skills from recorded demonstrations. (b) Online extraction of human behaviors through continuous interactions.

desired behaviors toward the goal, considering the resources, constraints, and setup. EA intelligent agents can learn offline from human-driven data (behavioral cloning, transfer learning, skill priors, and inverse reinforcement learning). They can also continuously exchange feedback with the demonstrator to be trained online by (direct policy learning) [15], [16], [17], [18]. The block diagrams in Fig. 2 summarize the differences between the two approaches.

Online learning has some flaws, including demanding a human-in-the-loop, which imposes extra cognitive burdens on the operator. It also needs metrics to evaluate the human decisions about the policy update, though it has yet to be matured. With offline training, there already exists data banks [19], [20] where meaningful task primitives and human behaviors can be extracted and transferred. These methods also generalize well to different task domains, making them robust against many intrinsic and extrinsic uncertainties [21], [22].

Although various methods have been developed to facilitate effective learning with limited task observations, few works have comprehensively summarized and contextualized these approaches within the domain of robotics. One such work in [23] reviewed different learning methods, ranging from imitation to reinforcement learning, for robot manipulation tasks. However, it lacks empirical benchmark studies that could provide useful insights into the performance and fitness of different discussed deep reinforcement learning algorithms across a diversified set of interaction tasks. In contrast, Saveriano et al. [24] presented a tutorial survey that provides rigorous mathematical formulations of existing dynamic MPs (DMPs). This survey systematically reviews the state-of-the-art literature on DMPs and is accompanied by an analysis of experimental results. Unlike their work, our approach is not intended to serve as a tutorial but to provide a general guideline and outlook that suits the audience with an understanding of different robot learning paradigms. While Fang et al. [25], Hussein et al. [26], and Zhu and Hu [27] have reviewed learning algorithms for robotic grasping, their focus was restricted to imitation learning, and they did not adequately address MP approaches that provide a micro-scale perspective on incremental learning strategies. Moreover, no works have comprehensively compared and contrasted MP and EA to the best of our knowledge. Therefore, this review paper aims to elucidate these two as the main doctrines of reducing the learning problem by demonstrating motions and skills to the learning agents. In addition, this survey contributes to the literature by presenting an empirical benchmarking of a robotic assembly task (peg-in-the-hole). Although, this benchmarking study differentiates our work from existing literature but it does not guarantee the universal behavior of such MP and EA approaches over other different tasks and their performance may vary based on task nature and its complexity. The main purpose of experimental results is to demonstrate how these algorithms behave under general industrial-graded assembly and we expect the same behavior under tasks of similar nature.

In summary, the following goals have been pursued.

1) Insights on MP and EA approaches and their potential use in robotic applications.
2) Empirical evaluation of the most common learning algorithms based on different performance metrics.
3) Time and space complexities of the highlighted approaches, their key characteristics, and use cases.
4) Identify research directions for MP and EA in imitation and reinforcement learning.

The remainder of this article is organized as follows. In Section II, an in-depth discussion on MP approaches and their realizations in encoding motion profiles and control laws is presented. Section III delves into the application and implications of exploiting distinct EA techniques to accelerate the iterative learning process, leading to the derivation of more optimal policies. In Section IV, a comprehensive analysis of the most promising evaluation metrics that have been used to benchmark the performance of the discussed MP and EA over the pick-place task is presented, along with an elaboration of the usability and applicability of each highlighted approach. Finally, Section V provides a summary of the core findings derived from the class of MP and EA approaches, followed by a proposal of promising future research directions for further exploration and exploitation.

II. Movement Primitives

Industrial robots mainly have three to six DoFs but in unstructured environments, they need as many DoFs as possible to generate a set of feasible solutions for their planning making their inference of the task and its constraints intractable [28]. Hence, a modular structure with simple mechanisms of known behaviors is better. In turn, well-tuned parallel or series organization of these basic motor control units (motions) tailors the motion for the given task, reducing both time and computation. Known as MPs, they were initially identified in animal studies [29], which pinpointed that the different regions in the neural circuit of a spinalized frog and their distinguishing functionality can be rearranged to form different movements.

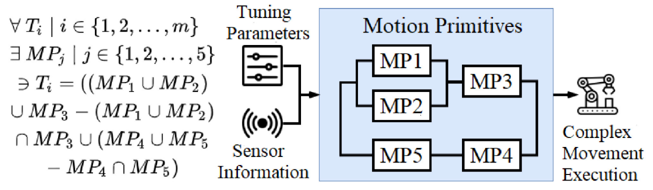

$\forall\, T_i \mid i \in \{1,2,\ldots,m\}$
$\exists\, MP_j \mid j \in \{1,2,\ldots,5\}$
$\ni T_i = ((MP_1 \cup MP_2)$
$\cup\, MP_3 - (MP_1 \cup MP_2)$
$\cap\, MP_3 \cup (MP_4 \cup MP_5$
$-\, MP_4 \cap MP_5)$
Tuning Parameters
Sensor Information
Motion Primitives
MP1
MP2
MP3
MP5
MP4
Complex Movement Execution

Fig. 3. General representation of motion primitives as a sequence of series and parallel combinations of discretized movements for complex tasks.

For MP, primitive motion blocks MP_j model an elaborate gesture or a complicated task T_i. As shown in Fig. 3, tuning parameters and sensory information are viewed as inputs. Depending on the nature of the task to be performed, several forms of MPs have been proposed in [23], [30], and [31].

A. Dynamic Movement Primitives

DMPs, a trajectory-based MP approach, are a deterministic abstraction of MP that model each moving target as a simple spring–damper system. DMP-based approaches were originally introduced in [32] and later modified in [33]. DMP models each MP as a combination of a well-specified stable dynamical system coupled with a nonlinear modulation term known as forcing function. The forcing function tries to bring the actual system as close as possible to a desired imaginary trajectory called the attractor field. Based on the task's specifications, there are two types of attractors [34]: 1) pointwise attractors for discrete tasks and 2) limit-cycle attractors for periodic movements. The dynamical system in DMP is described by

$$\begin{aligned}\tau\ddot{y} &= \alpha_y\big(\beta_y(g-y)-\dot{y}\big)+f\\ \tau\dot{z} &= \alpha_z(\beta_z(g-y)-z)+f^{1}\end{aligned}\tag{1}$$

where α and β are the constant gains, and g and y are the system's target and current states, respectively.

By default, the forcing function f is explicitly time-dependent, hampering the chance of coupling or coordinating with the remaining DoFs [33]. A first-order dynamical system ($\dot{x} = -\alpha_x x$ for discrete motion and $\tau\dot{\phi} = 1$ for rhythmic motion [33]), called Canonical system, implicitly encodes the temporal information to eliminate such time dependency of f. A phase variable x, defined by (2) and (3), respectively, for discrete and rhythmic movements, is assigned for this purpose. While $x = 1$ indicates the system's initial state, $x = 0$ marks the end of the task's execution. Furthermore, x acts as an activation node that terminates each task when it reaches to its end, making the system stable. Typically, a single canonical system is shared through a limb (e.g., a humanoid's arm) to coordinate DMP for multiple DoFs [33]

$$f(x) = \frac{\sum_{i=1}^{N}\Psi_i w_i}{\sum_{i=1}^{N}\Psi_i}x(g-y_0)\tag{2}$$

$$f(r,\phi)^{2} = \frac{\sum_{i=1}^{N}\Psi_i w_i}{\sum_{i=1}^{N}\Psi_i}r.\tag{3}$$

[1]Replacing y with z makes equation first order and more tractable.

[2]Periodicity is included in the choice of the canonical system but can be in the basis function as well [33]. For further understanding, refer to [32].

Equations (2) and (3) are canonized forcing functions of pointwise and periodic (rhythmic) attractors in the same order. Ψ_i and w_i are the ith basis function and weight $\forall i \in \{1, \ldots, N\}$.[3] In (3), r is the amplitude, and $\phi \in [0, 2\pi]$ is the phase angle for periodic oscillation.

As far as learning is concerned, different regression approaches like Gaussian mixture regression (GMR) [35] or Gaussian process (GP) [36] can be applied. In [32] and [33], a locally weighted regression (LWR) [37] is used. The features that need to be learned can be high-level parameters (τ, g, y_0, r), that scale up or down the task execution without changing its essence, or the weights w_i.

DMP has many appealing features. Besides simplicity in its implementation, it is flexible to adapt to new targets owing to its extrapolation capacity and the coordination it brings to the final movement. However, as a deterministic approach, DMP fails to consider uncertainties in target adaptation. Hence, it only returns a subset of the optimal solution (mean value) [38]. Since DMP follows the attractor field rather than the robot's actual trajectory, it only considers two via-points, the endpoints, ignoring the rest [39]. The state transition in DMP is prompt, which creates discontinuities in the velocity profile, leaving a jerk in its acceleration [8]. Finally, concurrent activation of distinct dynamical systems is impossible, hindering the multitasking capability of DMP [40].

To address these problems, DMP was modified by Hoffmann et al. [41] to give better modulation to new targets and become noise resilient. They derived an improved dynamical equation which is invariant to the choice of reference frame. The use of an acceleration term facilitates obstacle avoidance and reduces impulses compared to the initial version [32]. Apart from extending the use of DMP to include the transfer of skills from humans to the robot, Pastor et al. [42] also addressed the numerical issues faced by [41]. In addition, Stulp and Sigaud [43] minimized the noise effects on DMP's goal perturbation by subjecting the trajectory parameters (θ) to noise and calculating the corresponding cost function. In the end, incremental learning with path integral PI^2 updates the parameters (θ), which reduces the impact of noise on the trajectory. However, Meier and Schaal [44] followed a different strategy for reformulating the DMP to incorporate probabilistic implementation, encapsulating possible uncertainties.

B. Probabilistic Movement Primitives

Probabilistic MPs (ProMPs), the state-based MPs, are the probabilistic extension of DMP, which can adapt to varying starting and goal points. On top of that, ProMP modulates the via-points to avoid obstacles at a nonzero velocity profile. In its formulation, ProMP models the time-dependent variability of the trajectories using basis function representation. This representation is favored over the simple distribution of Gaussians as it is easier to train and get a time-continuous method [8]. Like DMP, ProMP utilizes squared-exponential (SE) basis functions representing discrete motions and Von Mieses basis functions for the rhythmic ones. Rather than driving just a subset of optimal solutions (mean value) in DMP [8], ProMP spans the whole training set for a range of solutions. The variability of the solution furnishes robust outcomes. At each time instant t, ProMP defines trajectory distribution based on the hierarchical Bayesian model (HBM). Its parameters determine the probability distribution of a specific trajectory stemming from its corresponding weights $p(\tau|w)$. The probability distribution of weights over variability parameters $\theta = \{\mu_w, \Sigma_w\}$ interactively captures the variance information $(p(w|\theta))$. At the end, marginalizing the weights out of trajectory distribution leads to the mentioned hierarchical architecture $p(\tau|\theta) = \int p(\tau|w) \times p(w|\theta) \times d(w)$.

Like DMP [32], [34], ProMP also eliminates the explicit time dependency by a phase variable z. ProMP reconstructs the motion and enhances its robustness by considering the areas with high variability. Variability allows ProMP to yield optimal solutions within the demonstration range. Moreover, ProMP can trigger multiple primitives concurrently (by Gaussian product of MPs) to couple the respective joints. Thus, it is possible to coordinate the movement within and between a robot's limbs (by covariance matrix). Activation function α enables a relatively smooth transition between two adjacent MPs to preclude harmful jerks' that can be seen in DMP [24].

ProMP modulates the new via-points by conditioning the MPs at each time step to reproduce the trajectory. It ensures the resultant trajectory passes through the via-point $p(w|x^*)$, where $x^* = [y_t^*, \Sigma_t^*]$ is the state where the robot should be at the given time t. y_t^* and Σ_t^* stand, respectively, for the position and required precision of via-points. The new states update the value of (μ_w and Σ_w) until the trajectory is assigned to the final via-point (the trajectory beyond the current via point is not optimized). This procedure iteratively continues prior to the final via-point being modulated. The optimal trajectory is synthesized in this state to meet all the via-points (the movement is completed). ProMP uses a single Gaussian in each iteration, like repetitive path decoding by GMR. Hence, it has a unitary prediction over the horizon, streamlining the GMM–GMR method [35]. Thus, it yields a substantially more accurate and general task recomposition compared to [35].

Since ProMP uses a single Gaussian in each iteration, it is impossible to accommodate more than one via-point at a time [8]. Like GMM, ProMP modulates the via-points with respect to the global frame. Purely stochastic in essence, ProMP is incapable of extrapolation [45], i.e., ProMP cannot modulate the arbitrary points located out of the demonstration set [39]. Besides, the conclusive result is bounded to a more extensive demonstration than the other MP-based methods to cover most of the possible scenarios; otherwise, ProMP is prone to error [39]. Eventually, ProMP needs the area with high variability in case of a single demonstration, the corresponding variance is zero, making it deterministic like DMP. Since ProMP needs to calculate the inverse of the covariance matrix, numerical errors may arise if the data is not sufficiently big. In this case, ProMP even underperforms DMP. Besides, ProMP again faces numerical error if the number of parameters to be learned is disproportionately big compared to the data set size [45]. It is worth mentioning that ProMP, either model-free or not, is not suitable for tasks that are loosely coupled in time [8].

[3] N is the total number of basis functions.

C. Parameterized Movement Primitives

Parametrized MPs (PMPs) encompass a fusion of trajectory-based and state-based approaches for encoding motion profiles. PMPs leverage the utilization of interpolation and spline functions or iterative nonlinear regulation to capture and represent the desired motions. Within this category, several notable methods used for robotic applications are discussed as follows.

1) Task Parameterized Models: One of the common hurdles in developing LfD is the extraction of consistent features from the human demonstrations [46], [47]. In response, the movements are parameterized in terms of projected trajectory into local frames $\{j\}$. Choosing local frames over the global one $\{O\}$ provides a better generalization over tasks. Although task parametrization is not necessarily twisted with the local frame adaptation, it is usually described as such. In fact, state-based approaches like original versions of mixture models (GMM and many variations of HMM) are also parameterizing the task through a neutral angle. Basically, what is usually known as task-parameterized (TP) methods refer to the upgraded version of their predecessors. TP models assign local frames to the landmarks, such as the relevant obstacles to the robot itself in the robot's workspace[4] [46]. TP models provide the robot with a detailed narrative and widen its vision of its environment and task features. Accordingly, making it easier to extract the shared features from the data set to adapt it efficiently to the task. To drive the task constraints (the common features), TP-based approaches are used to encode the desired reference trajectory from the demonstrations, as well as the precision and coordination prerequisite tracking under dynamic environments. However, TP model implementation is not simple and creates challenges in how and where to designate frames. This lack of a systematic strategy leads to more empirical approaches based on common sense.

A single Gaussian cannot capture data set for a task such as the movement of robot's end effector. As a result, a mixture of Gaussian distributions may apply, where each distribution locally captures a part of the whole. The mixture model is an umbrella term which includes GMR, HMM, Semi-HMM (SHMM), and others that stem from the same concept. Indeed, HMM can be considered a conditioned GMM [30].

2) Gaussian Mixture Model–Gaussian Mixture Regression: GMM–GMR pair is based on Gaussian distribution, which makes it probabilistic. It extracts the consistent tasks features and reproduces them in novel situations. To highlight the importance of the duo, most robotic tasks have stochastic traits. As a mixture model, GMM forms a linear superposition of Gaussian density functions [48], where the likelihood of the observation belonging to it is governed by: $p(X|\mu, \Sigma) = \sum_{k=1}^{K} \pi_k \mathcal{N}(X|\mu_k, \Sigma_k)$. Here, X is the observations and $\mathcal{N}(\cdot)$ is the Gaussian probability density function. $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ are K-component GMM's hyperparameters that stand for the prior probability, mean values, and covariance matrix of the kth components, respectively. Upon GMM delivering the model, GMR synthesizes the generalized movement. Given an arbitrary input as a new observation, GMR estimates the mean value $\hat{\mu}$ as generalized output (trajectory) and the task constraints and covariance matrix estimation $\hat{\Sigma}$.[5] In GMM–GMR's formulation, the data set consists of input and output $X = (X^{\text{In}}, X^{\text{Out}})$. GMM is brought by the joint probability distribution of input and output. However, GMR estimates the marginal probability distribution of output conditioned to new instance of input $\mathbb{E}(p(X^{\text{Out}}|X^{\text{In}}))$.

3) Hidden Markov Models: Based on Markov chains theory, a different realization, called HMM, was devised. First introduced in [49]. It has been widely used for voice recognition [50] ever since. In Bishop's book, a detailed description of its formulation and the advancements of HMM are provided [51].

GMM–GMR has many merits although it has problems for many scenarios. In its initial variants, GMM could not encapsulate the sequences of a task [52]. In [53], time was explicitly formulated (timestamp) to get the temporal modulation of the new instances of the task. Moreover, particularly in incremental learning, the demonstration of tasks should possess a certain level of resemblance [54]. Otherwise, the reassessment of the new tasks parameters is not going to be straightforward [45]. Consequently, it deprives GMM–GMR of extrapolating arbitrary instances of the task. Finally, in its basic format, one of the most notable flaws is its incapability of encoding the orientation [55]. To overcome this, Zeestraten et al. [56] reformulated GMM–GMR in the Riemannian space, where calculations are done locally in the planes, tangent to the manifold. Generally, the original GMM–GMR would be suitable for representing relevant features for continuous state spaces. However, instituting the transition between consecutive states and the duration to stay at each is also of importance. In essence, HMM falls into the mixture models category. Unlike GMM, HMM's states either proceed to the adjacent one or hold their ground. Such transitions between states are defined by their corresponding transition probabilities assigned to them. Since its development, HMM underwent modifications, e.g., SHMM [57] proposed to include the duration, which determines the time in each state. Overall, HMM tries to solve one of the following issues [50].

1) Probability of having specific observations given model.
2) Estimation of the most likely state sequence, given a set of observations and the model.
3) Determination of model's parameters, given state sequence, and corresponding observations.

For more information on other versions of HMM, refer to [51].

HMM, thanks to especially its Left-Right architecture,[6] intrinsically modulates time [58] to capture the inherent sequence in a task. Like most stochastic methods, HMM is suitable for tasks described by random probability distributions, such as teleoperation, manipulation, and locomotion. Since the uncertainties are included during its state estimation, HMM is resilient to their perturbations. Additionally, it can stochastically capture spatiotemporal variability [59]. HMM is a double stochastic process, which implies that its underlying hidden state takes another stochastic process to

[4]TP approaches are not limited to the frames, but in mainstream robotic applications, they are predominantly.

[5]Covariance matrix sets the required accuracy by the task as well.

[6]There are other implementations, but LR implementation is mainstream.

be visible [50]. Khokar et al. [60] took advantage of this feature (i.e., hidden states) to develop a teleoperation setup, which helps novice operators, as the system interprets the user's intention, to telemanipulate an object. The hidden state stands for the user's intention regarding the pregrasp poses.[7] Upon recognition, the system picks the intended object and then maps the corresponding pregrasp pose.

Nevertheless, HMM has its drawbacks. The discrete transition from one state to another leaves discontinuity in its velocity profile, which leads to a jerk in its acceleration contour. Therefore, Monte Carlo sampling and averaging techniques [61] are applied to smooth this out. Furthermore, HMM is not as good as GMR in task reconstruction [62]. In the end, like GMM, HMM cannot extrapolate every arbitrary input, limiting its strengths to the boundaries of the demonstration set.

4) Markov Chain Monte Carlo: Markov chain Monte Carlo (MCMC) has been utilized in various robot applications since the 1990s, initially implemented to solve the SLAM problem [63]. In essence, MCMC is founded on policy over priors, where the resultant posterior distribution with the highest reward is expected. It generates ergodic Markov chains $\{\theta^{(k)}\}_{k\geq 0}$, commencing from an initial estimate $\theta^{(0)}$. Subsequently, it proposes a new value from the target set, resulting in a sequence of samples from the proposal distribution probability distribution. Given the updated value $\theta^{(k-1)=\theta}$, the probability distribution is derived from the existing state θ' for the proposed distribution $\theta' \sim q(\theta'|\theta)$, which defines the exploration space for possible samples. Afterward, the algorithm calculates the acceptance ratio of the current and proposed samples $\rho(\theta,\theta') = ([q(\theta|\theta')]/[q(\theta'|\theta)])([\pi_n(\theta')]/[\pi_n(\theta)])$, where $\pi_n(\theta)$ is the distribution of interest. The algorithm accepts $\theta^{(k)} = \theta'$ the proposed value if the probability ratio, which measures the likelihood between the candidate and existing samples, is greater than or equal to 1. Otherwise, the proposed sample is accepted with a probability that depends on the acceptance ratio. MCMC is particularly useful when direct sampling of a complex distribution is difficult or impossible. This is especially true when the probability distribution is large, high-dimensional, and has complex dependencies between variables. MCMC can be used to encode trajectories without a closed-form solution, making it an effective tool for solving such problems.

An unbiased version of MCMC, which is more robust compared to the biased version, despite the computational costs in robust grasping tasks, is employed in [64]. Moreover, it has been benchmarked against well-studied stochastic gradient descent algorithms [65]. Simulation results demonstrate its superiority over the Gradient-based approach. Unlike its gradient-based counterpart, MCMC is noncontiguous, making it less prone to rendering locally optimal solutions in the presence of sudden changes in the gradient value. Furthermore, MCMC is model-free and does not require gradient calculation, making it more adaptable to different environments [64]. MCMC is one of the popular approaches in robot path planning. A multiobject optimal path from the Pareto-optimal trajectory space, called a Pareto-frontier solution set is proposed in [66]. The distribution over trajectories is implicitly uniform to ensure that the outcomes are representative and include the assigned objectives. Aghaei et al. [67] proposed and applied particle-MCMC (P-MCMC) to control the stabilization of a nonlinear system (inverted pendulum) with both quadratic and delayed rewards. Particle filters, which are Bayesian filter algorithms, are considered sequential Monte Carlo (SMC), run in each interaction to facilitate the MCMC with multiplication reward function. Compared to the additive one, multiplication reward functions are more rigorous toward error, making P-CMCM proposed here suitable for risk-sensitive tasks.

[7]The user's intention is not visible.

D. Via-Points Movement Primitives

DMP and ProMP both lacked functionalities, failing to deliver a generic solution for task reproduction. Zhou et al. [39] introduced via-points MPs (VMPs), a state-based MP approach, which built on previous work but also addressed DMP's ignoring of intermediary via points and ProMP's lack of extrapolation. VMP has an elementary trajectory h as its backbone and a shape modulation term f that gives the prior probability ω for the modulation, determining the likelihood of new point (during the learning stage).

Given the current values of parameter ω, the probability density function of an arbitrary via-point defines whether the point belongs to the data set or not. If the probability density of the new goal provided by ω exceeds the predefined threshold η, VMP modulates the via-point using the shape modulation term f. Otherwise, it extrapolates the via-point by translating the whole distribution into the new point's location. Afterward, the shape modulation term alters the trajectory's shape even further to bring it closer to the target.

Although improved by [41], [44], and [68], DMP does not explicitly consider the via-points properly mainly due to its philosophy, i.e., to track the actual trajectory and follow an attractor. Based on the results obtained by Zhou et al. [39], in the reconstruction of the alphabetical letters, the high variance regions do not have any significance. Consequently, the ProMP is more inclined to be erroneous than the VMP. Further, ProMP training takes more resources as it needs to cover the entire robot's workspace owing to its lack of extrapolation [8]. However, as mentioned before, VMP has no issues residing the points out of its training set.

E. Kernelized Movement Primitives

Kernelized MPs (KMPs), a trajectory-based MP technique, were introduced in [45]. When the dimension of input/output is high, the total number of the basis function rises sharply, making it not a straightforward problem to train. Hence, kernel functions relieve the direct usage of the basis functions in such scenarios. KMP, in its formulation, included extrapolation capacity to compensate for the neglected observation during the demonstration. This extends the robot's dexterity beyond the training set. Besides, KMP reconstructs a smoother trajectory compared to ProMP [45]. KMP suffers from high computational prices like many other of its peers. Moreover, tailoring and implementing KMP based on the task properties is problematic, restricting KMP from further development. More elaboration on the computational complexities of the kernel function in machine learning is presented in [69]. Driving KMP takes two essential elements: 1) probabilistic reference trajectory and 2) parametric trajectory.

From the training set, approaches like GMR [35] extract the probabilistic reference trajectory from a joint probability distribution. By assigning a weight vector w, the parametric trajectory can be obtained as a linear combination of the weighted basis function $\Theta^T \times w$, where $\Theta = \text{Blkdiag}[\phi_1, \ldots, \phi_B]$ is the matrix representation of basis functions. KMP minimizes the Kullback–Leibler (KL) Divergence loss to track the reference trajectory and meets the via-points. KL Divergence is well explained in [70] and [71]. KMP drives parameter variability into an optimization problem [45]. However, in ProMP [8], they are fulfilled by multiplication of known probability.

In general, two types of KMP exist based on the frame of reference: 1) local and 2) global. In global KMP, before the definition of the kernel $k(^{\cdot},^{\cdot})$, KMP needs to be initialized. Afterward, conditioning the parametric trajectory gives probabilistic reference trajectory. Finally, KMP delivers its prediction of μ, Σ, K, K^*, where they stand for mean, covariance matrix, kernel matrix ($K = [k(s_i, s_j)]\ \forall i, j \in \{1, 2, \ldots, N\}$), and kernel matrix corresponding to enquiry point S^* ($K^* = [k(s^*, s_i)]\quad \forall i \in \{1, 2, \ldots, N\}$) in the same order. These values solve two optimization subproblems for μ and Σ separately.

In local KMP, some reference frames are locally assigned considering the task features. Demonstrations and operations are all projected into the local frames to localize the task instances of the robot to increase its local movement range [45]. As discussed before, localization of the frames enhances the extrapolation span. Huang et al. [45] utilized the local version to augment dexterity in their setup.

Local KMP, like its standard version, can modulate a series of via-points. For modulation of a new query, points $m \in R^{M\times 1}$; local KMP starts with extending the reference trajectory to include new query points $D \in R^{N\times 1}$. Eventually, each via-point is modulated similar to a standard KMP. On top of all, KMP can blend different solutions (trajectories) for a task using the product of their reference trajectory distributions. Such superposition of trajectories brings about a weighted solution (weighted mean values and covariance matrix).

F. Functional Movement Primitives

Functional MPs (FMPs) are the group of trajectory-based and state-based MP approaches, that set a range of correspondences instead of a single time-dependent trajectory over the data points of demonstrations. The most popular among them are GPs and deep neural networks (DNNs).

1) Gaussian Processes: GPs, first mentioned in [72], are notable in this domain. Since GP is nonparametric, it has fewer hyperparameters to learn. Although its main applications are regression and classification, this article focuses only on regression. Details about classification are mentioned in [73]. GP conducts regression by defining a distribution over infinite possible functions that fit the given data set. As a result, it does not render only a subset (mean value) of all feasible solutions. After training, given the priors, GP regression (GPR) predicts (posteriors) to execute regression. The priors and their evaluations are ingrained using kernels (covariance). The algorithm iteratively updates priors until parameters converge to their optimal values [74].

GP's kernel is essential as it smoothens the resultant trajectories and helps extract the underlying function. However, when the dimension of features is disproportionately bigger than the data size, just like KMP [45], GP faces issues like KMP as described in (9). Moreover, picking the correct kernel is complicated yet crucial for guaranteed performance. The kernels can be a simple dot product ($|A| \cdot |B| \cdot \cos(\theta)$) or be harmonic. Due to the universal property of exponential function [74] and its convexity, SE kernels (4) is the default choice [75] in GP

$$k(x_i, x_j) = \sigma_f^2 \times \exp\left(-\frac{(x_i - x_j)^T (x_i - x_j)}{2l}\right). \tag{4}$$

σ and l are hyperparameters of the SE kernel to scale up or down the function vertically and horizontally and define the amount of data that can be presented. More information about the selection and tuning of hyperparameters is available in [75].

GP maps the input space into some virtual space in the real domain $\mathfrak{R}$. $f : \mathfrak{X} \rightarrow \mathfrak{R}$. Thus, (5) governs the marginal distribution over the function

$$\forall\ x_i \in \mathfrak{X}\ \exists\ p(f(x_1), \ldots, f(x_n)) = \mathcal{N}(m(x), K(x)) \tag{5}$$

where m and K are the mean function and covariance (kernel) function, respectively, defined by

$$\begin{aligned} \forall i, j \in &\{1, \ldots, n\}\ \exists\ \Sigma = \mathbb{E}\big((f(x_i) - m(x_i))\big(f(x_j) - m(x_j)\big)\big) \\ &= K(x_i, x_j) \in \mathbb{R}^{n\times n} \qquad (6) \\ \forall i\ &\in \{1, \ldots, n\}\ \exists\ m(x) = \mathbb{E}(f(\bar{x})) =\ \mu(x_i)\}\mu_{\in}\mathcal{R}^{n\times 1}. \end{aligned}$$

The two functions are the building blocks of SE kernels $f(x) \sim \mathcal{GP}(m, K)$. In the presence of i.i.d. noise $\varepsilon_i \sim \mathcal{N}(\cdot|0, \sigma^2)$, the outcome would be: $\forall i \in \{1, \ldots, n\} \exists\ \mathcal{D} = \{(X_i, f(X_i) + \varepsilon_i)\}$ GP predicts for a new goal, given a new arbitrary input and the data set $\mathcal{D}$ [76], defined by

$$p(y_{\text{new}}|x_{\text{new}}, D) = \int (y_{\text{new}}|x_{\text{new}}, f, D)p(f|D)df. \tag{7}$$

The predictive distribution is then determined using

$$\begin{aligned} &p(y_{\text{new}}|x_{\text{new}}, D) \sim \mathcal{N}\left(\mu_{\text{new}}, \sigma^2_{\text{new}}\right) \qquad (8) \\ &\mu_{\text{new}} = K_{\text{new},n}\left(K_n + \sigma^2 I\right)^{-1} y \\ &\sigma^2_{\text{new}} = K_{\text{new},new} - K_{\text{new},n}\left(K_n + \sigma^2 I\right)^{-1} K_{n,new}\sigma^2. \qquad (9) \end{aligned}$$

The inversion of $(K_n + \sigma^2 I)^{-1}$ term, in predictive distribution, is very expensive $\mathcal{O}(n^3)$ [77]. The higher level of complexity in GP, the more space it takes in the memory $\mathcal{O}(n^2)$ [78]. The other issue with GP, especially in its initial formulation, is its mostly stationary covariance function [79], usually when there are sudden changes in variance throughout the function. Henceforward, sparse approximations are used to reduce the computational complexity. Sparse approximations usually fall into three classes: 1) global sparse approximation; 2) local sparse approximation; and 3) hybrid approximation [77].

In global approximation, a few points called supports are selected, representing the observations in their vicinity. Consequently, the model is created by only a fraction of the whole data set $m << n$. A subset of the training set is substituted with m pseudo-inputs whose locations are assigned using gradient-based approximation. This approach reduces the complexity level to $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, respectively, for training and prediction [80]. Although the global approximation is swift in reducing the computational complexity, it cannot approximate the processes with fast dynamics (wiggly trajectories) well. Moreover, the assignment of the support points is not straightforward [81]. The local versions offer better robustness in the abstraction of the process while covering the weakness of the global one.

The data set is divided into smaller regions in local spars approximation [local GP (LGP)]. The gating network assigns a single GP, known as an expert, to its corresponding subspace of the whole task. Hence, the expert's contribution is limited to its surroundings. Eventually, the entire process is summarized by the superposition of all these experts along the path. This method is known as a mixture of experts (MEs) [82]. LGP uses the point in its surroundings, making prediction easier [77]. In [83], 3 m GPs are assigned, where one is responsible for the experts, and the other two are used to model the noises and a separate gating network. Since the whole data set is used for the computations, the overall complexity would be $\mathcal{O}(n^3)$, which is even more computationally expensive. Contrary, Rasmussen and Ghahramani [79] divided the whole trajectory into m equal regions and then assigned an expert for each. As a result, the complexity level dropped to $\log(\mathcal{O}(n^3))$.

To bypass the flaws of assigning experts for each task, they have introduced the concept of infinite experts accompanied by a gating network using the Dirichlet process. Nguyen-Tuong and Peters [84] examined the accuracy and efficiency of LGPs regarding their computational time. Afterward, they validated their method on a real-time model-based controller of a fast-compliant robot. In short, LGP is applied to more extensive sample-sized data with higher variation around its adjacent data points. Besides, LGP is a suggested approach for online learning [84].

Ultimately hybrid solutions were presented in [77] to take advantage of both domains and address the nonstatic abstraction challenge that the GP faces. In the implementation of the heteroscedastic GP, the noise rate is proportional to the input during regression. Accordingly, whilst a GP is responsible for capturing the noise-free process, the other GP is designated for the noise variance [85]. Hence, pseudo-input can improve the performance of GP for nonstatic systems to some extent [80].

2) Deep Neural Networks: DNNs are a type of FMP that are characterized by their overparameterization relative to nonparametric GPs. DNNs optimize their parameters by backpropagating i.i.d. priors, which makes them effective nonlinear function approximators. They are particularly well suited for large and dynamic data sets, as they have a computational complexity that rises linearly and can handle new data points without the need for function refitting [86]. In fact, a single-layer fully connected neural network with an i.i.d. prior over its parameters is equivalent to a GP in the limit of infinite network width [87]. This allows for exact Bayesian inference on regression tasks by evaluating the corresponding GP. By inducing a hidden layer with infinite width and deriving based on Bayesian marginalization over intermediate layers, uncertainty can be taken out to make DNNs more interpretable. One approach to addressing the limitations of GPs while improving the efficiency and quality of DNNs is the development of deterministic DNNs, such as neural network GP (NNGP) [88]. NNGP has been used for various robotic tasks, including robust grasping [89] motion planning [90], and task orchestration [91].

In robot learning, Learning from Observation (LfO) is a subset of the LfD paradigm where labeled examples are used to train the system from the demonstrations. Particularly in robotics, an algorithm analyses the expert's movements, devises them into a sequence of states, then maps each to their corresponding action. Although it is an effective approach, it is important to consider the challenges it entails. First, the state–action mapping between the learner and the expert should be fast yet accurate. Besides, the mentioned map may not be feasible in the presence of two structurally different agents (learner and expert). This issue is fundamental when a robot tries to clone a human operator. In this regard, Soans et al. [92] proposed a DNN architecture to accurately recognize the states and their corresponding actions. Contrary to the ad-hoc methods, whose applications are limited to a particular task, it recognizes the underlying patterns of images and videos. In regards to processing the visual sensory stream, [93] applied a two-layered hierarchical DNN architecture to predict the manoeuvrable profile of the robot in its environment to avoid collision by tracking the via-points. The approach is less prone to convergence failures during the optimization process due to mechanical limitations, nonlinear path/dynamic constraints, and nonconvex collision avoidance constraints. Since it is trained offline using pregenerated optimized trajectory observations, it facilitates online path planning that iterates to compute the optimal path.

III. Experience Abstraction

EA incrementally encodes new task-related behaviors using prior knowledge to speed up the learning process. It is either about agents learning from their own experiences or exploiting others' behavior to accommodate learning [94] efficiently over time. Besides getting overfitted to the task, Self-exploration takes longer to train. Hence, the actual task variables initialize the agent instead of random values. However, a proper balance between initialization and iterative encoding needs to be made to achieve the desired behavior. In general, the agent can be either a value-based or policy-based algorithm. Agents would train by either evaluating the values of state–action pair $Q(S, A)$ or optimal policy π^* to select the effective actions [95]. Nonetheless, both are trying to solve a similar problem formulated by the Markov decision process (MDP). An MDP for incremental learning consists of a tuple with five elements {States (S), Actions (A), Rewards (R), Transitions (ρ), Discounts (γ)}.

According to MDP, the agent tries to maximize the cumulative discounted reward to perform the given task optimally. To obtain greater discounted returns, the value-function/policy-network needs to be appropriately configured and initialized. The three well-known techniques to achieve it are: 1) transfer learning; 2) behavioral cloning (BC); and 3) priors-over-skills [96].

A. Skill Transfer

Skill Transfer is an approach where a pretrained agent is either extended to learn a new task by using existing model parameters or shares its trained configuration with other inexperienced agents for continual learning.

Considering the sampling inefficiency of learning in complex environments, the learned skills for one task are often transferred to others using different mechanisms, which improves learning efficacy. One common option concerning transfer skills would be initializing new policy parameters with learned skills. Subsequently, the hyperparameters would be tuned to adjust the learned task to the new task [42]. However, its application would be limited to scenarios where there are resemblances between the original and the new tasks. Otherwise, a subset of skills that retain semantic information is exploited [97]. This type of transfer learning is substantially used by value-based agents, where the abstraction of state-space ignores irrelevant observations to task skills (distractors) and then generalizes to state spaces of different sizes. In stochastic environments and continuous complex state spaces, DNNs are used to abstract the given value function within the simulated environment. In this regard, domain randomization is used and adapted to a physical system within the bounded set of policy parameters [98]. Besides, the neural networks prioritize tracking the history of the state, action, and goal tuples over the state–action pairs. In this case, the latter only applies if it helps leverage the policy parameters across multiple goals [99].

However, instead of transferring skills in a single instance, it is sometimes helpful to consider the problem as sequential learning under multiple instances. For instance, when the pretrained neural network is used for different tasks, its weights will be frozen during training. After that, a few lateral output layers will be added per task requirements, which speeds up the learning process, reduces the computational burden, and improves the inference performance as well [100]. In addition, sequential learning is applicable where it is more expensive to explore than exploit for complex tasks. Therefore, it starts initially with a sparse representation of tasks. In an incremental learning paradigm, the agent gradually builds intuition using smooth gradient steps with positive rewards [101].

The commonly used algorithms based on this idea are as follows.

1) *Guided Policy Search [102]:* Deriving interpolation controllers for task instances and using them for generating data to train a policy network.
2) *Hindsight Experience Replay (HER) [103]:* Fake play between the agent-state and hindsight-observation for a given task to end being in goal.
3) *Scheduled Auxiliary Control [104]:* Choosing from the hierarchy of predefined task actions and learning to reach a defined goal on-fly.

B. Behavioral Cloning

In response to the exploration–exploitation dilemma of incremental learning, BC was introduced. BC initializes the robot's policy, so it takes greedy actions from the state space at early iterations. In this regard, BC falls into the supervised learning paradigm to learn policy parameters, either as a state-to-action mapping $\pi : S \rightarrow A$ for deterministic or a state–action pair mapping of probabilities over actions: $\pi : S \times A \rightarrow R$ stochastic policies. Besides allowing the agent to mimic the behavior of the demonstrator in novel conditions, this approach helps to infer the reward function that the agent is trying to optimize [105] for inverse reinforcement learning.

Several studies have shown BC's efficacy in many robotic tasks while initializing incremental learning policies [106], [107], [108]. The earlier research in [109] and [110] focused on using BC to understand model dynamics and reward functions from human demonstrations, which fosters learning optimal policies with better convergence and sample efficiency. Meanwhile, Kim et al. [111] applied BC during the policy iteration phase for value function approximation using constrained convex optimization. Whereas, in [112], [113], and [114], BC merged with motion planning using probabilistic inference was furnished to learn, detect, and recover from conflicting conditions. Moreover, it employed perceptual modalities, i.e., vision and tactile to scale learned behaviors in complex scenarios.

In this context, Hester et al. [115] and Nair et al. [116] have introduced new algorithms called deep Q-learning from demonstrations (DQfD) and deep deterministic policy gradient from demonstration (DDPGfD). These approaches sought to learn policies for Atari games (DQfD) and object manipulation tasks (DDPGfD). The n-step Q-learning loss function was applied to fulfill the Bellman equation's optimality. For gradient-free agents, however, the policy optimization from demonstrations (POfD) algorithm [117] included expert skills. Accordingly, POfD minimizes the gap between the demonstrated and learned policies with adversarial learning objectives and sparse reward signals. The demonstration augmented in policy gradient (DAPG) approach [118] used human demonstrations for gradient-based agents. Demonstration deduces a BC loss as a pretraining step. Thus, applying weighted heuristic optimization bears an extended cost function. This cost function interpolates BC and policy gradient losses to train an optimal policy using the natural policy gradient technique [119].

C. Priors Over Skills

Encoding priors over skills is a way to extract behaviors and skills from data without any reward or task information. These data are generated either by humans or agents interacting with their surroundings. They are used to solve the downstream tasks, where the action space of trained policy represents the set of extracted skills. With an increasing number of such skills, the time–space complexity of action rises. The resulting complexities compromise learning performance due to the need for enormous extrapolation within the space of derived skills [120]. This technique also gave rise to promising results for offline incremental learning settings, where the policy is deduced from the previous experience of the agent. Eventually, it was exploited to accelerate online incremental learning [121]. However, the downside of this approach is the need to annotate the collected experience for inferring and leveraging the desired skills across multiple tasks.

Considering the time–space complexity of such intertask skill transfer methods, the stochastic latent variables are used to define priors over the skills. Thus, the need for extended policy blocks for representing more skills would be alleviated [122], [123]. These types of skill embedding facilitate a faster exploration of intelligent agents within the complex skill space. Additionally, they are applied to avoid overestimating action values $Q(s, a)$, which eventually accelerates incremental learning for downstream tasks. These intelligent agents represent asymmetric masking of behaviors across the library of skills that enables generalization within encoded priors over skills [124]. Besides, this approach helps to overcome the issue of biases of an intelligent agent toward the learned behaviors and exploit skills for lifelong learning [125].

IV. Discussion

This article studies some of the most important MPs and EA approaches and uses the standard peg-in-hole task as a benchmark to evaluate their performances. The metrics detailed in the following are exploited to evaluate each method.

A. Evaluation Metrics

1) Time–Space Complexity: Time–space complexity is a paradigm for investigating the computational performance of an algorithm. It is governed by the time taken and memory consumed by an algorithm to execute the given set of instructions. The time and space constraints of an algorithm are competing interests; hence, a tradeoff should be made between its efficiency (i.e., quicker and correct outputs) and finiteness (i.e., terminate and avoid memory leaks [126]). The asymptotic notations are used to compare and evaluate the time–space complexity of multiple algorithms, especially the Big-O notation ($\mathcal{O}$). It measures the change in the performance of two algorithms by increasing or decreasing the input sequence size. It defines the worst case scenario of an algorithm. Mathematically, it is used to determine an upper bound of an algorithm. For instance, if an algorithm is defined by $f(N) = \mathcal{O}(g(N))$, for some positive constants K and M, the resultant function would be $f(N) \leq K \times g(N)\ \forall\ N \geq M$. In this case, In this case, knowing how $f(N)$ increases as the input N size increases are essential.

2) Root Mean-Squared Error: Root mean-squared error (rMSE) is a simple metric to evaluate the performance of prediction models; it is dependent solely on the current observation. It measures the average Euclidean difference between predicted and target values. rMSE penalizes even the small differences by squaring them, which leads to overestimating the trained model performance [127]. It is defined as

$$\text{rMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(Y_{\text{pred}_t} - Y_{tgt_t}\right)^2}$$

where N is the number of samples, Y is model output, and t is the current instance.

3) Coefficient of Determination: The Coefficient of determination (R^2) defines the variation in predicted values compared to targets. R's range varies between 0 and 1, and the closer it gets to 1, the closer it captures 100% of variance in the targets. Unlike rMSE, by adjusting a number of independent variables, R^2 is less prone to be affected by the outliner and false indications, which overfits the model to the data [128]. Contrary to the rMSE, R^2 highlights the correlation between the model's parameter and the data. Hence, it can modify the model's parameters based on the upcoming data. Mathematically, it is expressed as

$$R^2 = 1 - \left(\frac{N-1}{N-K-1}\right)\left(\frac{\sum_{i=1}^{N}\left(Y_{\text{act}} - Y_{\text{pred}}\right)^2}{\sum_{i=1}^{N}\left(Y_{\text{act}} - Y_{\text{mean}}\right)^2}\right)$$

where N is the number of samples, K is the number of independent variables, and Y is output with respective actual, predicted, and mean values.

4) Information Loss: Information theory measures both the amount of information present in a given data set and that lost by the model after training to evaluate the difference between the predicted and actual values the cross-entropy loss is commonly used in classification problems [51]. Such losses can also be directly inferred based on the rules of information theory. the entropy of a trained model or proposed algorithm [129]. Model's entropy denote is denoted as

$$H(X) = -\mathbb{E}\big[\log p(X)\big] \quad \& \quad p(X) = f_1(X) f_2(X) \ldots f_n(X)$$

$p(X)$ is the probability distribution of random variable X, which is the output of the trained model. Each function component $f_i(X)$ is independent and asymmetrically contributes to the information gained from $p(X)$. The greater the model's entropy $H(X)$, the greater the information gained from data.

It can be concluded that the nature of data varies for different ranges of robot tasks, including both prehensile and nonprehensile motions, and as a result, the usefulness and measures of these evaluation metrics also vary. Smaller values of rMSE and *IL* indicate greater accuracy and precision for a given learning algorithm to reproduce the task at hand. Conversely, higher values of R^2 indicate error-free capture of the variability in the data and greater proximity of the regenerated solution from the chosen algorithm to its reference trajectory. For time–space complexity ($\mathcal{O}$), it is advisable

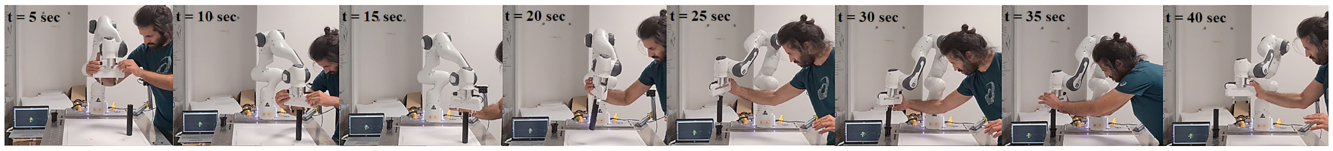


Fig. 4. Human subject guiding physically a redundant robot arm to execute a peg-in-hole task. The corresponding 3-D Cartesian space positional data is collected for training and evaluating different discussed models.

to select algorithms that exhibit small metric values during training and inference stages, by taking into account the hardware capabilities, the nature of task, and the working environment.

B. Algorithms for Robotic Tasks

Table I summarizes the functional aspects, computational intricacies, and robotic applications of various approaches from the MPs and EA classes. From an application standpoint, DMP has demonstrated promising results in contact-rich manipulation and location tasks due to their second-order control law encoding that enables the exploitation of robots' dynamic behavior at the torque level. On the other hand, PMP approaches, which are discretized and nonparametric in nature, are generally preferred for tasks that require nonprehensile motion or intuitive HRC without physical contact. ProMP, VMP, and KMP share similarities in that they are probabilistic approaches that use linear quadratic regulator to track reference trajectory distribution and can easily incorporate environmental descriptors, such as via and end-points. Consequently, these approaches are commonly used for tasks that require impulsive forces for instant momentum generation or human-centric motion and path planning. In contrast, FMP leverages their ability to select the appropriate function to govern the existing state environment. Consequently, FMPs are commonly used for long-horizon tasks, as they can linearly compose the trajectory, or for problems that require frequent switching of contact states.

On the other hand, within the realm of EA, BC stands out as the most commonly used method for many complex robotic tasks, due to its ability to initialize either the policy or value function from demonstrations. Its popularity is primarily attributed to its simplicity and local task exposure. In practice, BC is often used in conjunction with data-aggregation tools to leverage off-policy data to accelerate the exploration process for finding locally optimal actions to accomplish a given task within a defined time frame. However, recent studies have shown that the PoS strategy outperforms BC in terms of effectiveness, as it includes both the actions and their effects on the learning process. This approach is particularly well suited for tasks that involve precision grasping or asymptotically stabilized locomotion, as it leverages spatial encoding and dynamic sequencing to generate effective plans. In contrast, skill transfer techniques often suffer from high bias and variance problems and can result in a significant drop in performance when encoding a targeted task. Moreover, successful skill transfer across changing task conditions typically requires an additional domain adaptation and task embedding module. Therefore, skill transfer is best suited for tasks that involve sequential coordination or a high degree of repetition accuracy.

C. Experimental Benchmarking

In the framework of LfD, the underlined methods were applied to reconstruct a standard peg-in-hole task (Fig. 4), which gives an insight into their performance. Withstanding its simplicity, the peg-in-hole involves all the substantial exchanges between robot and object, including grasping, exploration, and manipulation. The statistical outcomes for all indexes are presented in Table II.

In this case study, an Intel i5 6500U @ 2.3-GHz system with 16-GB RAM and NVIDIA GTX 1650 Ti 4 GB was utilized for training and testing the algorithms. Six different human demonstrations with varying initial conditions were collected. MP algorithms were applied using their predefined settings from the peers. The proximal policy optimization (PPO) technique was used for EA. PPO was chosen for multiple reasons, including faster convergence, better sampling efficiency, and ease of tuning. Henceforward, PPO has been widely applied to different manipulation tasks [130], [131], [132], [133].

As shown in Table II, out of MP methods, DMP relatively outperforms the others as DMP ensures not only the adaptive behavior for varying task conditions but also inherits key characteristics from ergodic control for better exploration. This task starts with grasping a peg from the top, bringing it all the way to the hole by following the desired path. Finally, explore the hole with the bottom of the peg to align and then insert it. Although the results achieved by VMP are also promising but since the given task does not involve explicit superposition and modulation of conflicting trajectories for intermediary and final goal points so it eventually underperforms the DMP.

In the case of EA, the trained PPO agent tries to take correlated consecutive steps to remain within the trusted region. Although being robust against environmental noise and task uncertainties during exploration minimizes the entropy, but it requires frequent tuning of hyper-parameters and, if not properly configured, suffers from instability in reaching the task goal (i.e., properly positioning the peg into the hole). It was also observed that during certain trajectory rollouts, the PPO misses the target (hole) and left the object (peg) sideways, which was primarily due to its poor policy initialization.

It must be noted that; the smaller the values of rMSE and *IL* indices, the better the task reproduction of the tested algorithm becomes. Conversely, for R^2, it is preferable to have a higher value to regenerate it closer to the task's base solution (kinaesthetic trajectory). Although none of the single

TABLE I
SUMMARY OF DISCUSSED TECHNIQUES IN TERMS OF THEIR KEY CHARACTERISTICS AND COMPUTATIONAL COMPLEXITY

Method list	**Attributes**				
	Keywords	***Pros***	***Cons***	***Complexity***	***Applications***
Movement Primitives					
DMP	• Deterministic; • Parametric; • Supervised; • Generative; • Non-linear differential dynamics; • Imaginary attractor tracking; • No explicit time dependency; • Uni-modal distributions.	• Time independent; • Scaling flexibility; • Stability; • Needs only a single demonstration.	• Lack of Via-point modulation possibility; • It is not multitasking; • Discontinue inter-primitive transition; • Missing robust performance in perturbation; • Takes only mono-dimensional temporal inputs .	$\mathcal{O}(n)$	• Interactive rehabilitation [134]; • Contact-aware Locomotion [11]; • Prehensile Motions [135] [136]; • Context-driven Path traversing [137] [138];
PMP	• Probabilistic; • Parametric; • Supervised; • Generative; • Actual trajectory tracking; • Weighted sum of Gaussians; • Bayesian Inference.	• Input can be temporal or spatial; • Controlled transition and duration to each state; • Better task convergence;	• Discontinuity on intermediate points; • Requires more demonstrations; • More hyper-parameters to tune.	$\mathcal{O}(n^2)$	• Household manipulations [139] [140]; • human-robot collaboration [141] [142]; • Guesture recognition [143] [144]; • Stable walking [145], adaptive manipulations [146];
ProMP	• Probabilistic; • Parametric; • Supervised; • Generative; • Interaction of single Gaussian; • Actual trajectory tracking; • Weighted sum of Gaussianas; • No explicit time dependency; • Multi-modal distributions; • Linear quadratic regulator.	• Multitasking; • Smooth inter-primitive transition; • Robust towards uncertainties; • Spatiotemporal synchronization.	• No extrapolation possibilities; • Multiple nodes can not be abstracted at the same time ; • Numerical issues if the input features vector is not proportional to the observation size; • It requires an inclusive dataset for a proper via-point adaptation;	$\mathcal{O}(log(n))$	• Co-manipulation of objects and assembly [147]; • Playing stroke games [38] [148];
VMP	• Probabilistic; • Parametric; • Supervised; • Generative; • Polynomial basis function; • Multi-modal distributions; • Bayesian inference.	• Extrapolation capacity is included; • A straightforward implementation that takes an MP and a few via-point to capture the task; • Better accuracy;	• Obstacle avoidance is not as expected. • Controller stability is compromised. • Local spare representation makes it prone to noise.	$\mathcal{O}(nlog(n))$	• Human-centric grasping, path traversing [39];
FMP	• Probabilistic; • Parametric; • Supervised; • Generative; • RBF kernels; • Bayesian inference; • Distribution over functions; • Multi-modal distributions.	• More accurate regression; • Simple math makes it easy to implement. • Better fit without cross validation.	• Usually it is computationally expensive. • Requires more information for prediction • Compromises efficiency in high dimensional space.	$\mathcal{O}(n^3)$	• Trajectory tracking and reaching [36]; • bipedal Locomotion [149]; • robust environment interaction [150];
KMP	• Probabilistic; • Parametric; • Supervised; • Generative; • Tracks actual trajectory; • Bayesian inference; • No direct basis function.	• Better extrapolation possibilities; • Requires lesser demonstration, specially compared with ProMP; • Deals with dimensionality issues of input with high number of features.	• Usually it is computationally expensive; • The accuracy of the approach highly bounds with the type of kernel; • In the absence of proper prior information on the target position, trajectory adaptation will face problems.	$\mathcal{O}(n^3)$	• Object transportation, Human-robot collaboration [45] [151]; • Dexterous Manipulation [152] [153]; • Non-holonomic motion planning [154];

(Continued)

approaches can give the best results for all three metrics, the most promising ones are from DMP for MPs and BC and Priors over Skills from EA.

V. CONCLUSION AND RECOMMENDATIONS

In this review article, we summarized the key traits, strengths, limitations, computational complexity, and

TABLE I
(Continued) SUMMARY OF DISCUSSED TECHNIQUES IN TERMS OF THEIR KEY CHARACTERISTICS AND COMPUTATIONAL COMPLEXITY

Method list	**Attributes**				
	Key feature	***Pros***	***Cons***	***Complexity***	***Applications***
Experience Abstraction					
Skill Transfer	• Deterministic; • Parametric; • Supervised; • Discriminating; • Uni and multi-modal.	• Alleviate the need for extensive training; • Suitable for dealing with an exploration-exploitation dilemma; • Less prone to task overfitting;	• Sparse representation affects task accuracy. • Kinematic-dynamic constraints on the mapping between agents. • Applicable only to tasks having similar initial and final conditions.	$\mathcal{O}(n)$	• Object location [155] [156]; • In-hand manipulation [157] [158]; • Dual-arm dexterous manipulations [159] [160];
Behavioural Cloning	• Deterministic; • Parametric; • Supervised; • Discriminating; • Reward function optimization.	• Incorporate human intuition in executing a task; • Leads to achieve optimal control; • Simple but effective strategy for direct policy learning;	• Data need to be independent and identically distributed. • Treats decision making process as prediction problem. • Less generalizable to task with varying dynamics.	$\mathcal{O}(nlog(n))$	• Furniture assembly [114]; • Environment exploration [161]; • Autonomous locomotion [162].
Priors Over Skills	• Deterministic & probabilistic; • supervised & unsupervised; • parametric & non-parametric; • Bayesian inference.	• Distributed nature makes it easier to adapt. • Generalized behaviour for global stability. • Similar performance for both policy and value-based agents.	• Need for latent representation. • Requires more resources. • Necessitates sdditional higher dimensional skill embedding.	$\mathcal{O}(n^2)$	• Object grasping and stacking [122]; • Navigation, precision grasping, fine manipulation [121];

TABLE II
EXPERIMENTAL EVALUATION OF DIFFERENT MPS AND EA ALGORITHMS USING STATISTICAL MEASURES

Method	$rMSE$	R^2	IL
DMP	0.13876	0.26625	**1.02854**
PMP	0.30452	0.40449	1.13224
ProMP	0.27532	0.20976	1.84252
VMP	0.12043	0.27352	1.21465
FMP	0.22132	0.24742	1.57432
KMP	0.19751	0.25314	1.34211
PPO-ST	0.07632	0.39015	1.18764
PPO-BC	**0.02324**	0.41783	1.09852
PPO-PoS	0.05431	**0.42132**	1.18653

applications of different robot learning algorithms ranging from motion primitives to skill embedding. We addressed both soft and rigid boundaries of various agents under distinct circumstances that affect their learning performance in realizing specific and generalized motions, skills, and actions. A standard peg-in-hole task was reconstructed and evaluated using different statistical metrics to benchmark and understand the performance of discussed approaches. Based on what we have found, kernel-based and policy-gradient-based approaches for LfD and Incremental Learning outperformed the others in the same order. However, they suffer from computational loads when applied to more complex tasks.

Because of the limited applications and hidden potential of algorithms for task decomposition, trajectory encoding, motion representation, shared behaviors, task semantic masking, and skill embedding discussed in this article, the following recommendations are made for potential future directions.

1) *Affordance Encoding:* Instead of learning actions that lead to task goals through demonstration, it is more desirable to encode the relationship between the actions taken by an agent (such as a robot) and their effects on the environment over long-horizon trajectories. This relationship, known as the affordance, not only helps to learn about the interaction features required to achieve task goals but also about how tasks evolve with the subsequent motions executed by the agent [163]. By learning affordance, the agent can leverage previous knowledge about environmental objects and tools to better perform tasks in new contexts. In this domain, state-based MPs approaches have significant potential to extract desired features from data sets about candidate objects and tools that the agent needs to interact with, such as whether they are graspable or nongraspable, avoid collision or exploit extrinsic dexterity, and so on.
2) *Multimodal Cloning:* In the field of LfD, the majority of techniques rely on a single modality to enable robots to acquire the knowledge required to perform a desired task. This knowledge typically consists of a sequence of actions, associated skills, and contextual information related to the execution of the task. However, this approach can limit the robot's ability to incorporate semantic information and adapt to environmental changes [164]. As a result, there is a need to explore multiple modalities, including natural

language instructions (leveraging the emergence of GPT models [165]), visual observations (using skeleton tracking SKDs [166]), haptic feedback (via wearables [167]), and kinaesthetic teaching, to facilitate the robot's ability to learn generalized task goals, motion plans, and motor actions that can be adapted to distinct task conditions.

3) *Discriminative Modeling:* The majority of techniques utilized in LfD research makes the assumption that data is i.i.d. or preprocessed to enable effective learning of useful task semantics and temporal coordination. However, this assumption is restrictive as it implies that most demonstrations are performed by domain experts. Consequently, such approaches are limited to laboratory conditions where researchers are skilled in demonstrating the required manipulation or location task to robots using appropriate interfaces. Nevertheless, algorithms based on this assumption suffer from generating suboptimal policies when demonstrations are either heterogeneous or partially executed [168]. Thus, there is a pressing need for algorithms that can quantify and model missing information and uncertainties in the demonstrations and environmental dynamics and make robust and adaptive decisions under uncertain conditions.

4) *Hierarchical Decision Making:* In most of the LfD approaches, the primary objective is to learn the motion profile for a dexterous robot system to reproduce a given task in a more optimal and generalistic manner. However, while these approaches have made significant strides in achieving this objective, they have not fully realized the potential for learning task sequences and motion plans simultaneously using the concept of dynamic behavior trees [169]. Dynamic behavior trees enable the learning controller to arrange the hierarchy of subtasks in a discrete space and plan continuous actions among these discrete subgoals in a more optimal manner, taking into account the task description and given data. This approach can achieve the best performance on overall task execution under different contexts, considering both long-term task goals as necessary conditions and short-term objectives as sufficient constraints. Moreover, this approach can obtain more generalized and adaptive behaviors for solving a class of tasks that share common features, i.e., bin-picking, handling, and sorting [170].

5) *Transceive Learning:* Transferring knowledge from a trained model to its naive counterpart for better exploitation and less exploration is promising to curtail the computation burden and improve performance. However, in specific scenarios, like sim-to-real skill transfer, where there is not enough correspondence between their workspaces, this approach may fall into deficiencies. Instead of randomly initializing the domain parameters in this condition, it is highly recommended to use real-time environment data using different perceptual feedback and train the naive agent (digital twin). The trained policy will guarantee better domain adaptation and generalize well to different task extensions, similar to the skill transfer in real-sim-real adaptation (digital twin) [171].

References

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[2] W. Si, N. Wang, and C. Yang, "A review on manipulation skill acquisition through teleoperation-based learning from demonstration," *Cogn. Comput. Syst.*, vol. 3, no. 1, pp. 1–16, 2021.

[3] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3828–3834.

[4] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," in *Springer Handbook of Robotics*. Cham, Switzerland: Springer, 2016, pp. 1995–2014.

[5] T. Lozano-Perez, "Robot programming," *Proc. IEEE*, vol. 71, no. 7, pp. 821–841, Jul. 1983.

[6] S. Muench, J. Kreuziger, M. Kaiser, and R. Dillman, "Robot programming by demonstration (RPD)-using machine learning and user interaction methods for the development of easy and comfortable robot programming systems," in *Proc. Int. Symp. Ind. Robots*, vol. 25, 1994, p. 685.

[7] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Proc. 11th Int. Symp. Robot. Res.*, 2005, pp. 561–572.

[8] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Auton. Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[9] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 44–54, Jun. 2010.

[10] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[11] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robot. Auton. Syst.*, vol. 47, nos. 2–3, pp. 79–91, 2004.

[12] L. Rozo, S. Calinon, D. Caldwell, P. Jiménez, and C. Torras, "Learning collaborative impedance-based robot behaviors," in *Proc. AAAI Conf. Artif. Intell.*, vol. 27, 2013, pp. 1422–1428.

[13] E. A. Rückert, G. Neumann, M. Toussaint, and W. Maass, "Learned graphical models for probabilistic planning provide a new class of movement primitives," *Front. Comput. Neurosci.*, vol. 6, p. 97, Jan. 2013.

[14] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions," *J. Math. Psychol.*, vol. 85, pp. 1–16, Aug. 2018.

[15] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 1–8.

[16] B. D. Ziebart et al., "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[17] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 49–58.

[18] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots, "Fast policy learning through imitation and reinforcement," 2018, *arXiv:1805.10413*.

[19] S. Dasari et al., "RoboNet: Large-scale multi-robot learning," 2019, *arXiv:1910.11215*.

[20] S. Cabi et al., "Scaling data-driven robotics with reward sketching and batch reinforcement learning," 2019, *arXiv:1909.12200*.

[21] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," 2019, *arXiv:1910.11956*.

[22] N. Y. Siegel et al., "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning," 2020, *arXiv:2002.08396*.

[23] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 1395–1476, 2021.

[24] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," 2021, *arXiv:2102.03861*.

[25] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *Int. J. Intell. Robot. Appl.*, vol. 3, pp. 362–369, Sep. 2019.

[26] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, “Imitation learning: A survey of learning methods,” *ACM Comput. Surveys*, vol. 50, no. 2, pp. 1–35, 2017.
[27] Z. Zhu and H. Hu, “Robot learning from demonstration in robotic assembly: A survey,” *Robotics*, vol. 7, no. 2, p. 17, 2018.
[28] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, “Imitation learning of dual-arm manipulation tasks in humanoid robots,” *Int. J. Humanoid Robot.*, vol. 5, no. 2, pp. 183–202, 2008.
[29] R. Shadmehr and F. A. Mussa-Ivaldi, “Adaptive representation of dynamics during learning of a motor task,” *J. Neurosci.*, vol. 14, no. 5, pp. 3208–3224, 1994.
[30] S. Calinon and D. Lee, *Learning Control*. Dordrecht, The Netherlands: Springer, 2017.
[31] Y. Huang, J. Silvério, L. Rozo, and D. G. Caldwell, “Generalized task-parameterized skill learning,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 5667–5474.
[32] S. Schaal, “Dynamic movement primitives—A framework for motor control in humans and humanoid robotics,” in *Adaptive Motion of Animals and Machines*. Tokyo, Japan: Springer, 2006, pp. 261–280.
[33] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.
[34] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots,” in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, 2002, pp. 1398–1403.
[35] S. Calinon and A. G. Billard, “What is the teacher’s role in robot programming by demonstration? Toward benchmarks for improved learning,” *Interaction Studies*, vol. 8, no. 3, pp. 441–464, 2007.
[36] D. Forte, A. Gams, J. Morimoto, and A. Ude, “On-line motion synthesis and adaptation using a trajectory database,” *Robot. Auton. Syst.*, vol. 60, no. 10, pp. 1327–1339, 2012.
[37] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning,” *Artif. Intell. Rev.*, vol. 11, pp. 11–73, Feb. 1997.
[38] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 1–9.
[39] Y. Zhou, J. Gao, and T. Asfour, “Learning via-point movement primitives with inter-and extrapolation capabilities,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2019, pp. 4301–4308.
[40] A. Pervez and D. Lee, “Learning task-parameterized dynamic movement primitives using mixture of GMMs,” *Intell. Service Robot.*, vol. 11, no. 1, pp. 61–78, 2018.
[41] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, “Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 2587–2592.
[42] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, “Learning and generalization of motor skills by learning from demonstration,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 763–768.
[43] F. Stulp and O. Sigaud, “Path integral policy improvement with covariance matrix adaptation,” 2012, *arXiv:1206.4621*.
[44] F. Meier and S. Schaal, “A probabilistic representation for dynamic movement primitives,” 2016, *arXiv:1612.05932*.
[45] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, “Kernelized movement primitives,” *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 833–852, 2019.
[46] S. Calinon, D. Bruno, and D. G. Caldwell, “A task-parameterized probabilistic model with minimal intervention control,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 3339–3344.
[47] F. Guenter, M. Hersch, S. Calinon, and A. Billard, “Reinforcement learning for imitating constrained reaching movements,” *Adv. Robot.*, vol. 21, no. 13, pp. 1521–1544, 2007.
[48] E. Pignat and S. Calinon, “Bayesian Gaussian mixture model for robotic policy imitation,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4452–4458, Oct. 2019.
[49] L. Rabiner and B. Juang, “An introduction to hidden Markov models,” *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, Jan. 1986.
[50] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
[51] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4. Cham, Swizerland: Springer, 2006.
[52] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, “Statistical dynamical systems for skills acquisition in humanoids,” in *Proc. 12th IEEE/RAS Int. Conf. Humanoid Robots (Humanoids)*, 2012, pp. 323–329.
[53] S. Calinon, P. Kormushev, and D. G. Caldwell, “Compliant skills acquisition and multi-optima policy search with EM-based reinforcement learning,” *Robot. Auton. Syst.*, vol. 61, no. 4, pp. 369–379, 2013.
[54] C. Sylvain, *Robot Programming by Demonstration: A Probabilistic Approach*. Lausanne, Switzerland: EPFL Press, Aug. 2009.
[55] S. Calinon, “Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control,” *IEEE Robot. Autom. Mag.*, vol. 27, no. 2, pp. 33–45, Jun. 2020.
[56] M. J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, “An approach for imitation learning on Riemannian manifolds,” *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1240–1247, Jul. 2017.
[57] X. D. Huang and M. A. Jack, “Semi-continuous hidden Markov models for speech signals,” *Comput. Speech Lang.*, vol. 3, no. 3, pp. 239–251, 1989.
[58] L. Rozo, P. Jiménez, and C. Torras, “Force-based robot learning of pouring skills using parametric hidden Markov models,” in *Proc. IEEE 9th Int. Workshop Robot Motion Control*, 2013, pp. 227–232.
[59] R. Dillmann, “Teaching and learning of robot tasks via observation of human performance,” *Robot. Auton. Syst.*, vol. 47, nos. 2–3, pp. 109–116, 2004.
[60] K. Khokar, R. Alqasemi, S. Sarkar, K. Reed, and R. Dubey, “A novel telerobotic method for human-in-the-loop assisted grasping based on intention recognition,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 4762–4769.
[61] A. M. Schmidts, D. Lee, and A. Peer, “Imitation learning of human grasping skills from motion and force data,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 1002–1007.
[62] M. J. Zeestraten, S. Calinon, and D. G. Caldwell, “Variable duration movement encoding with minimal intervention control,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016, pp. 497–503.
[63] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” *Mach. Intell. Pattern Recognit.*, vol. 5, pp. 435–461, 1988.
[64] V. T. Aghaei, A. Ağababaoğlu, S. Yıldırım, and A. Onat, “A real-world application of Markov Chain Monte Carlo method for Bayesian trajectory control of a robotic manipulator,” *ISA Trans.*, vol. 125, pp. 580–590, Jun. 2022.
[65] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.
[66] J. Lee, D. Yi, and S. S. Srinivasa, “Sampling of Pareto-optimal trajectories using progressive objective evaluation in multi-objective motion planning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 1–9.
[67] V. T. Aghaei, A. Onat, and S. Yıldırım, “A Markov Chain Monte Carlo algorithm for Bayesian policy search,” *Syst. Sci. Control Eng.*, vol. 6, no. 1, pp. 438–455, 2018.
[68] A. D. Dragan, K. Muelling, J. A. Bagnell, and S. S. Srinivasa, “Movement primitives via optimization,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 2339–2346.
[69] N. Cesa-Bianchi, Y. Mansour, and O. Shamir, “On the complexity of learning with kernels,” in *Proc. Conf. Learn. Theory*, 2015, pp. 297–325.
[70] S. Kullback, *Information Theory and Statistics*. New York, NY USA: Courier, 1997.
[71] S. Kullback and R. A. Leibler, “On information and sufficiency,” *Ann. Math. Stat.*, vol. 22, no. 1, pp. 79–86, 1951.
[72] C. Williams and C. Rasmussen, “Gaussian processes for regression,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 8, 1995, pp. 1–7.
[73] J. Hensman, A. Matthews, and Z. Ghahramani, “Scalable variational Gaussian process classification,” in *Proc. Artif. Intell. Stat.*, 2015, pp. 351–360.
[74] J. Wang, “An intuitive tutorial to Gaussian processes regression,” 2020, *arXiv:2009.10862*.
[75] D. Duvenaud, “Automatic model construction with Gaussian processes,” Ph.D. dissertation, Dept. Comput. Sci., Univ. Cambridge, Cambridge, U.K., 2014.
[76] Z. Ghahramani, “A tutorial on Gaussian processes (or why I don’t use SVMS),” in *Proc. MLSS Workshop Talk Gaussian Processes*, 2011, pp. 1–8.
[77] E. Snelson and Z. Ghahramani, “Local and global sparse Gaussian process approximations,” in *Proc. Artif. Intell. Stat.*, 2007, pp. 524–531.
[78] J. Quinonero-Candela and C. E. Rasmussen, “A unifying view of sparse approximate Gaussian process regression,” *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, Dec. 2005.

[79] C. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001, pp. 881–888.

[80] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 18, 2005, pp. 1257–1264.

[81] M. Schneider and W. Ertel, "Robot learning by demonstration with local Gaussian process regression," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 255–260.

[82] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.

[83] V. Tresp, "Mixtures of Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 13, 2000, pp. 1–7.

[84] D. Nguyen-Tuong and J. Peters, "Local Gaussian process regression for real-time model-based robot control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 380–385.

[85] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard, "Most likely heteroscedastic Gaussian process regression," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 393–400.

[86] J. Strohbeck, J. Müller, M. Herrmann, and M. Buchholz, "Deep kernel learning for uncertainty estimation in multiple trajectory prediction networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2022, pp. 11396–11402.

[87] B. He, B. Lakshminarayanan, and Y. W. Teh, "Bayesian deep ensembles via the neural tangent kernel," in *Proc. NeurIPS*, vol. 33, 2020, pp. 1010–1022.

[88] G. Pang, L. Yang, and G. E. Karniadakis, "Neural-net-induced Gaussian process regression for function approximation and PDE solution," *J. Comput. Phys.*, vol. 384, pp. 270–288, May 2019.

[89] S. Ottenhaus, D. Renninghoff, R. Grimm, F. Ferreira, and T. Asfour, "Visuo-haptic grasping of unknown objects based on Gaussian process implicit surfaces and deep learning," in *Proc. IEEE-RAS 19th Int. Conf. Humanoid Robots (Humanoids)*, 2019, pp. 402–409.

[90] Y. Zhang and Z. Zhang, *Repetitive Motion Planning and Control of Redundant Robot Manipulators*. New York, NY, USA: Springer, 2013.

[91] J. Lee, J. Feng, M. Humt, M. G. Müller, and R. Triebel, "Trust your robots! Predictive uncertainty estimation of neural networks with sparse Gaussian processes," in *Proc. Conf. Robot Learn.*, 2021, pp. 1168–1179.

[92] N. Soans, E. Asali, Y. Hong, and P. Doshi, "SA-Net: Robust state-action recognition for learning from observations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 2153–2159.

[93] R. Chai, H. Niu, J. Carrasco, F. Arvin, H. Yin, and B. Lennox, "Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 10, 2022, doi: 10.1109/TNNLS.2022.3209154.

[94] X. Zhou, H. Wu, J. Rojas, Z. Xu, and S. Li, "Incremental learning robot task representation and identification," in *Nonparametric Bayesian Learning for Collaborative Robot Multimodal Introspection*. Singapore: Springer, 2020, pp. 29–49.

[95] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[96] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv:2005.01643*.

[97] R. Platt, C. Kohler, and M. Gualtieri, "Deictic image mapping: An abstraction for learning pose invariant manipulation policies," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8042–8049.

[98] K. Bousmalis et al., "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 4243–4250.

[99] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.

[100] A. A. Rusu et al., "Progressive neural networks," 2016, *arXiv:1606.04671*.

[101] M. Svetlik, M. Leonetti, J. Sinapov, R. Shah, N. Walker, and P. Stone, "Automatic curriculum graph generation for reinforcement learning agents," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, 2017, pp. 2590–2596.

[102] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search (2015)," 2015, *arXiv:1501.05611*.

[103] M. Andrychowicz et al., "Hindsight experience replay," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[104] M. Riedmiller et al., "Learning by playing solving sparse reward tasks from scratch," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4344–4353.

[105] P. Abbeel and A. Y. Ng, *Inverse Reinforcement Learning*, Stanford Univ., Stanford, CA, USA, 2010.

[106] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Stat.*, 2011, pp. 627–635.

[107] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," 2018, *arXiv:1805.01954*.

[108] Y. Duan et al., "One-shot imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1087–1098.

[109] S. Schaal, "Learning from demonstration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1996, pp. 1–8.

[110] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *Proc. ICML*, vol. 97, 1997, pp. 12–20.

[111] B. Kim, A.-M. Farahmand, J. Pineau, and D. Precup, "Learning from limited demonstrations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 2859–2867.

[112] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Skill generalization via inference-based planning," in *Proc. RSS Workshop Math. Models Algorithms Human–Robot Interact.*, 2017, pp. 1–8.

[113] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Adv. Robot.*, vol. 25, no. 5, pp. 581–603, 2011.

[114] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 131–157, 2015.

[115] T. Hester et al., "Deep *Q*-learning from demonstrations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 3223–3230.

[116] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 6292–6299.

[117] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2469–2478.

[118] H.-R. Lee and T. Lee, "Improved cooperative multi-agent reinforcement learning algorithm augmented by mixing demonstrations from centralized policy," in *Proc. 18th Int. Conf. Auton. Agents Multiagent Syst.*, 2019, pp. 1089–1098.

[119] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor–critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[120] T. Shankar, S. Tulsiani, L. Pinto, and A. Gupta, "Discovering motor programs by recomposing demonstrations," in *Proc. Int. Conf. Learn. Rep.*, 2019, pp. 1–9.

[121] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," 2020, *arXiv:2010.11944*.

[122] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *Proc. Int. Conf. Learn. Rep.*, 2018, pp. 1–9.

[123] C. Lynch et al., "Learning latent plans from play," in *Proc. Conf. Robot Learn.*, 2020, pp. 1113–1132.

[124] Y. Lee, S.-H. Sun, S. Somasundaram, E. S. Hu, and J. J. Lim, "Composing complex skills by learning transition policies," in *Proc. Int. Conf. Learn. Rep.*, 2018, p. 9.

[125] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2052–2062.

[126] G. J. Woeginger, "Space and time complexity of exact algorithms: Some open problems," in *Proc. Int. Workshop Parameterized Exact Comput.*, 2004, pp. 281–290.

[127] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Res.*, vol. 30, no. 1, pp. 79–82, 2005.

[128] D. J. Ozer, "Correlation and the coefficient of determination," *Psychol. Bull.*, vol. 97, no. 2, p. 307, 1985.

[129] J. C. Baez, T. Fritz, and T. Leinster, "A characterization of entropy in terms of information loss," *Entropy*, vol. 13, no. 11, pp. 1945–1957, 2011.

[130] A. Rajeswaran et al., "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," 2017, *arXiv:1709.10087*.

[131] G. Garcia-Hernando, E. Johns, and T.-K. Kim, “Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 9561–9568.

[132] T. Yoneda, C. Schaff, T. Maeda, and M. Walter, “Grasp and motion planning for dexterous manipulation for the real robot challenge,” 2021, *arXiv:2101.02842*.

[133] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto, “Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation,” 2022, *arXiv:2203.13251*.

[134] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Trajectory formation for imitation with nonlinear dynamical systems,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* , vol. 2, 2001, pp. 752–757.

[135] A. Colomé and C. Torras, “Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes,” *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 602–615, Jun. 2018.

[136] T. Zhao, M. Deng, Z. Li, and Y. Hu, “Cooperative manipulation for a mobile dual-arm robot using sequences of dynamic movement primitives,” *IEEE Trans. Cogn. Develop. Syst.*, vol. 12, no. 1, pp. 18–29, Mar. 2020.

[137] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, “Upper-body kinesthetic teaching of a free-standing humanoid robot,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3970–3975.

[138] A. Ude, A. Gams, T. Asfour, and J. Morimoto, “Task-specific generalization of discrete and periodic dynamic movement primitives,” *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 800–815, Oct. 2010.

[139] S. Kim, R. Haschke, and H. Ritter, “Gaussian mixture model for 3-DoF orientations,” *Robot. Auton. Syst.*, vol. 87, pp. 28–37, Jan. 2017.

[140] A. K. Bozcuoğlu, Y. Furuta, K. Okada, M. Beetz, and M. Inaba, “Continuous modeling of affordances in a symbolic knowledge base,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2019, pp. 5452–5458.

[141] L. Rozo, S. Calinon, and D. G. Caldwell, “Learning force and position constraints in human–robot cooperative transportation,” in *Proc. IEEE 23rd Int. Symp. Robot Human Interact. Commun.*, 2014, pp. 619–624.

[142] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell, “Learning optimal controllers in human–robot cooperative transportation tasks with position and force constraints,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2015, pp. 1024–1030.

[143] J. Kwon and F. C. Park, “Natural movement generation using hidden Markov models and principal components,” *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1184–1194, Oct. 2008.

[144] D. Lee, C. Ott, and Y. Nakamura, “Mimetic communication model with compliant physical contact in human–humanoid interaction,” *Int. J. Robot. Res.*, vol. 29, no. 13, pp. 1684–1704, 2010.

[145] L. Rozo, P. Jiménez, and C. Torras, “Robot learning from demonstration of force-based tasks with multiple solution trajectories,” in *Proc. IEEE 15th Int. Conf. Adv. Robot. (ICAR)*, 2011, pp. 124–129.

[146] A. K. Tanwani and S. Calinon, “Learning robot manipulation tasks with task-parameterized semitied hidden semi-Markov model,” *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 235–242, Jan. 2016.

[147] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, “Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks,” *Auton. Robots*, vol. 41, no. 3, pp. 593–612, 2017.

[148] E. Rueckert, J. Mundo, A. Paraschos, J. Peters, and G. Neumann, “Extracting low-dimensional control variables for movement primitives,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 1511–1518.

[149] A. Werner, D. Trautmann, D. Lee, and R. Lampariello, “Generalization of optimal motion trajectories for bipedal walking,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2015, pp. 1571–1577.

[150] J. Schreiter, P. Englert, D. Nguyen-Tuong, and M. Toussaint, “Sparse Gaussian process regression for compliant, real-time robot control,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 2586–2591.

[151] J. Silvério, Y. Huang, F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, “Uncertainty-aware imitation learning using kernelized movement primitives,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2019, pp. 90–97.

[152] S. Katyara, F. Ficuciello, D. G. Caldwell, B. Siciliano, and F. Chen, “Leveraging kernelized synergies on shared subspace for precision grasping and dexterous manipulation,” *IEEE Trans. Cogn. Develop. Syst.*, early access, Sep. 6, 2021, doi: 10.1109/TCDS.2021.3110406.

[153] S. Katyara, F. Ficuciello, F. Chen, B. Siciliano, and D. G. Caldwell, “Vision based adaptation to kernelized synergies for human inspired robotic manipulation,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 6491–6497.

[154] N. Deng, Y. Cui, S. Zhang, and H. Li, “Autonomous vehicle motion planning using kernelized movement primitives,” in *Proc. IEEE Int. Symp. Netw. Comput. Commun. (ISNCC)*, 2021, pp. 1–6.

[155] H. Karaoguz and P. Jensfelt, “Object detection approach for robot grasp detection,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 4953–4959.

[156] T. Weng, A. Pallankize, Y. Tang, O. Kroemer, and D. Held, “Multi-modal transfer learning for grasping transparent and specular objects,” *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 3791–3798, Feb. 2020.

[157] B. Bocsi, L. Csató, and J. Peters, “Alignment-based transfer learning for robot models,” in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2013, pp. 1–7.

[158] T. E. Lee, J. A. Zhao, A. S. Sawhney, S. Girdhar, and O. Kroemer, “Causal reasoning in simulation for structure and transfer learning of robot manipulation policies,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 4776–4782.

[159] D. Seita et al., “Deep transfer learning of pick points on fabric for robot bed-making,” in *Proc. Int. Symp. Robot. Res.*, 2019, pp. 275–290.

[160] M. Schwarz et al., “Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 3347–3354.

[161] M. W. Kadous, C. Sammut, and R. Sheh, “Behavioural cloning for robots in unstructured environments,” in *Proc. Adv. Neural Inf. Process. Syst. Workshop*, 2005, pp. 1–8.

[162] S. Desai, I. Durugkar, H. Karnan, G. Warnell, J. Hanna, and P. Stone, “An imitation from observation approach to transfer learning with dynamics mismatch,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 3917–3929. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/28f248e9279ac845995c4e9f8af35c2b-Paper.pdf

[163] H. Min, C. Yi, R. Luo, J. Zhu, and S. Bi, “Affordance research in developmental robotics: A survey,” *IEEE Trans. Cogn. Develop. Syst.*, vol. 8, no. 4, pp. 237–255, Dec. 2016.

[164] X. Lu et al., “3D tactile based object recognition for robot hands using force-sensitive and bend sensor arrays,” *IEEE Trans. Cogn. Develop. Syst.*, early access, Oct. 17, 2022, doi: 10.1109/TCDS.2022.3215021.

[165] “GPT-4 passes the bar exam.” Mar. 2023. [Online]. Available: https://ssrn.com/abstract=4389233

[166] S. Secil and M. Ozkan, “Minimum distance calculation using skeletal tracking for safe human–robot interaction,” *Robot. Comput. Integr. Manuf.*, vol. 73, Feb. 2022, Art. no. 102253.

[167] J. Bimbo, C. Pacchierotti, M. Aggravi, N. Tsagarakis, and D. Prattichizzo, “Teleoperation in cluttered environments using wearable haptic feedback,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 3401–3408.

[168] Q. Li and Y. Wang, “A novel teacher-assistance-based method to detect and handle bad training demonstrations in learning from demonstration,” *IEEE Trans. Cogn. Develop. Syst.*, vol. 14, no. 3, pp. 948–956, Sep. 2022.

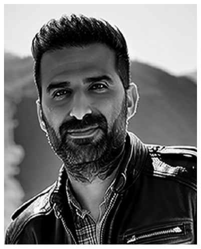
[169] J. Li, Z. Li, X. Li, Y. Feng, Y. Hu, and B. Xu, “Skill learning strategy based on dynamic motion primitives for human–robot cooperative manipulation,” *IEEE Trans. Cogn. Develop. Syst.*, vol. 13, no. 1, pp. 105–117, Mar. 2021.

[170] C. R. Garrett et al., “Integrated task and motion planning,” *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, pp. 265–293, May 2021.

[171] Y. Liu, H. Xu, D. Liu, and L. Wang, “A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping,” *Robot. Comput. Integr. Manuf.*, vol. 78, Dec. 2022, Art. no. 102365.

Mehrdad Tavassoli received the B.Sc. degree in automation and control from Hamedan University of Technology, Hamadan, Iran, in 2013, and the M.Sc. degree in mechatronics engineering from the Polytechnic University of Turin, Turin, Italy, in 2019. He is currently pursuing the Ph.D. degree in robotics with the University of Pisa, Pisa, Italy.

His research interests include soft robotics and machine learning, focusing on using machine learning techniques to facilitate fast and reliable abstraction of manipulation tasks in the framework of learning from demonstration paradigms.



Sunny Katyara (Member, IEEE) received the Ph.D. degree in advanced robotics from the University of Naples Federico II, Naples, Italy, in 2022.

He is a Senior Robotics Researcher with Irish Manufacturing Research, Dublin, Ireland, where he leads technical activities within Robotics and Automation Team. He has more than five years' experience of working at various research centers and academic institutes. He has published more than 20 research articles in top-notched journals and conferences. His research focuses on human-inspired fine manipulation, reactive human–robot collaboration, and deep reinforcement learning.

Dr. Katyara has also won the Best Research Paper Awards at 12th International HFR and IEEE ICRA-VITAC workshops.

Maria Pozzi (Member, IEEE) received the B.S. degree (cum laude) in computer and information engineering, the M.S. degree (cum laude) in computer and automation engineering, and the Ph.D. degree (cum laude) in information engineering and science from the University of Siena, Siena, Italy, in 2013, 2015, and 2019, respectively.

She is an Assistant Professor with the University of Siena. Her main research interests include robotic grasping, simulation and modeling of soft robots, haptic interfaces for human–robot collaboration, and educational robotics.

Dr. Pozzi won the RAS Haptics Grant promoted by the IEEE RAS Technical Committee on Haptics in 2018. In 2021, she was selected as one of the RSS Pioneers and she was an Invited Speaker at the IFRR Global Robotics Colloquium on The Future of Robotic Manipulation. Since 2023, she has been an Associate Editor of *IEEE Robotics and Automation Magazine*.

Nikhil Deshpande (Member, IEEE) received the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 2012, investigating intelligent autonomous navigation, leveraging interaction with wireless sensor networks.

He is a Researcher and the Head of the Vicarios Mixed Reality and Simulations Lab, Italian Institute of Technology, Genoa, Italy. His research interests focus on improving situational awareness for human users in advanced telerobotics systems, utilizing holistic approaches in mixed reality and robot learning. Previously, he focused on the design, development, and evaluation of novel surgical devices, surgeon–machine telesurgery interfaces, and controllers for robot-assisted surgeries.

Darwin G. Caldwell (Fellow, IEEE) received the B.Sc. and Ph.D. degrees in robotics from the University of Hull, Hull, U.K., in 1986 and 1990, respectively, and the M.Sc. degree in management from the University of Salford, Salford, U.K., in 1994

He is the Founding Director of the Italian Institute of Technology, Genoa, Italy, where he is also the Director of the Department of Advanced Robotics. He has pioneered developments in compliant and variable impedance actuation, soft and human friendly robotics, and the creation of "softer," safer robots, that draw on developments in materials, mechanisms, sensing, actuation, and software. This has been fundamental to advances in humanoids, quadrupeds, medical robotics, and exoskeletons. Key robots developed by his team include iCub; COMAN; WALK-MAN; the HyQ series of quadrupeds (HyQ, HyQ2Max, and HyQ-Real); the PHOLUS Centaur robot, and exoskeletons, such as the XoSoft, XoTrunk, XoShoulder, and XoElbow. In addition, he has worked extensively in surgical and rehabilitation robotics, where his teams have developed systems, such as the computer-aided laser microsurgery system, the Cathbot, Cathbot-Pro, and SVEI (for catheterization and tissue-type detection), and the Arbot (ankle rehabilitation robot). He is or has been an Honorary Professor with the University of Manchester, Manchester, U.K.; the University of Sheffield, Sheffield, U.K.; Bangor University, Bangor, U.K.; King's College London, London, U.K.; and Tianjin University, Tianjin, China. He has published over 700 papers, has over 25 patents, and has received over 50 awards/nominations at international conferences and events.

Prof. Caldwell is a Fellow of the Royal Academy of Engineering.

Domenico Prattichizzo (Fellow, IEEE) received the M.S. degree in electronics engineering and the Ph.D. degree in robotics and automation from the University of Pisa, Pisa, Italy, in 1991 and 1995, respectively.

In 1994, he was a Visiting Scientist with MIT AI Laboratory, Cambridge, MA, USA. Since 2009, he has been a Scientific Consultant with the Istituto Italiano di Tecnologia, Genoa, Italy. He is currently a Professor of Robotics with the University of Siena, Siena, Italy. He is the Leader of a research unit in several EU projects. He is also the Co-Ordinator of an EU ECHORD-EXPERIMENT and an EU IP Collaborative Project. He has authored more than 2000 articles in his research field. His main research interests include haptics, grasping, visual servoing, mobile robotics, and geometric control.

Dr. Prattichizzo was a recipient of the IEEE 2009 Chapter of the Year Award. He was the Vice Chair of the Special Issues of the IEEE Technical Committee on Haptics from 2006 to 2010. He was the Chair of the Italian Chapter of the IEEE RAS from 2006 to 2010. He has been the Chair of the IEEE RAS Early Career Awards Evaluation Panel since 2013.