Robot Motor Skill Transfer with Alternate Learning in Two Spaces

Jian Fu, Xiang Teng, Ce Cao, Zhaojie Ju, Ping Lou

***Abstract*— Recent research achievements in Learning from Demonstration (LfD) demonstrate that the reinforcement learning is effective for the robots to improve its movement skills. The current challenge mainly remains in how to generate new robot motions, which have similar preassigned performance indicator but are different from the demonstrated tasks. To deal with the above issue, this paper proposes a framework to represent the policy and conduct imitation learning and optimization for robot intelligent trajectory planning, based on the improved local weighted regression(iLWR) and policy improvement with path integral by dual perturbation (PI^2-DP). Besides, the reward-guided weight searching and basis function's adaptive evolving are performed alternately in two spaces, *i.e.* the basis function space and the weight space, to deal with the above problem. The alternate learning process constructs a sequence of two-tuples which joins the demonstration task and new one together for motor skill transfer. So that the robot skills can be gradually learnt from similar tasks, and those skills can also correspond the demonstrated tasks to dissimilar tasks in different criterion. Classical via-points trajectory planning experiments are performed with the SCARA manipulator, a 10 DOF planar and the UR robot. These results show that the proposed method is not only feasible but also effective.**

***Index Terms*— motor skill acquisition, alternate learning in two spaces(ALTS), improved local weighted regression(iLWR), PI^2-DP**

I. Introduction

HUMAN motor skills are ideal examples for robot movement skill generation. Robotic researchers put enormous efforts into studying human motor skills and trying to transfer such advanced motor skills into robot motions. Recently, reinforcement learning has been successfully integrated into the LfD framework and provides an effective way for robots not only to learning skills from human demonstrations but also to be able to self-improve such skills to complete more complex tasks than human demonstration. The reinforcement learning based LfD (LfDRL) becomes one hottest topic in robotics and currently receive significant research interest [1]–[8].

We in the paper focus on scenarios with a few demonstration for a motion planning task, so we only investigate the dynamic model scheme instead of end-to-end training scheme. Generally speaking, the LfDRL has 3 phases to develop: representation phase, imitation phase and optimization phase. The first phase is to select a parametric policy representation to efficiently modulate online and exhibit robustness via a dynamic model. Two of the most popular dynamic models are Dynamical Movement Primitives (DMPs) [9]–[11] and Motion Scheduling (DS) [12], [13] . In the former, the robot motions are modelled as superimposition of a linear dynamic system and a nonlinear term. The latter not only represents motion planning in the form of a nonlinear autonomous dynamic system but also guarantees the global convergence. In the second phase, the various models are designed to learn appropriate parameters based on the demonstrated data, such as radial basis function networks [14], regularized kernel least-square, locally weighted regression (LWR) [15], [16] and Gaussian process [17], [18] etc.

In the optimization phase, model-free policy search methods are more popular than model-based method since learning a policy is normally easier than modelling a robot in a dynamic environment. Some classic model-free policy search methods were proposed recently, such as Policy learning by Weighting Exploration with the Returns (PoWER), Cost-Regularized Kernel Regression (CRKR), Covariance Matrix Adaptation-Evolutionary Strategy (CMA-ES), Trust Region Policy Optimization (TRPO) and path integral with policy improvement(PI^2). As for PoWER [19], a linear policy parameters rather than action are perturbed and a reward-weighted regression is conduced later. CRKR is a nonparametric nonlinear policy method which is is more flexibility to model a function than by using specified feature vectors [20], [21]. However, its output dimensions of the policy are typically modeled as independent Gaussian distributions ignoring the correlations. Differently, CMA-ES is a black-box optimizer. It applies heuristic information to estimate the weight and update the exploration distribution, which is found to be efficient in practice but not well founded theoretically [22]. TRPO [23] is a method for optimizing control policies with guaranteed monotonic improvement by means of advantage function. However, it adopts policy representation in the form of stochastic neural network without decomposing task into optimal control and supervised phases [24]. PI^2 [25] is a promising method which can acquire optimal control laws for nonlinear continuous time systems by performing monte-carol rollouts to solve the HJB equation indirectly.

It is noticed that many current studies focus on the contribution in one phase. Besides, there is an open dilemma–it can not endow the robot flexible and specific imitation capability

This work was supported by the National Natural Science Foundation of China under Grants 61773299, 51575412, Excellent Dissertation Cultivation Funds of Wuhan University of Technology under Grants 2017-YS-066, 2017-YS-067. Corresponding Author: Zhaojie Ju

J. Fu, X. Teng and C. Cao are with the School of Automation, Wuhan University of Technology, Wuhan, Hubei 430070, China. (e-mail: fujian@whut.edu.cn, tengxiang_131@163.com, 1310518618@qq.com).

Z. Ju is with School of Computing, University of Portsmouth, UK. (e-mail: zhaojie.ju@port.ac.uk).

P. Lou is with the School of Information, Wuhan University of Technology, Wuhan, Hubei 430070, China. (e-mail: louping@whut.edu.cn).

simultaneously in traditional LfDRL framework. In our previous research [26], [27], an effective policy representation has been developed and a preliminary study on alternate learning in two spaces has been carried out. In this paper we aim to develop a generic method with theoretic analysis for motor skill learning, with which robot could be broad applicable for the motion task with various performance indicators compared with the demonstration and of good quality for given motion planning task as well. Inspired by the idea of deep learning [28], [29] which conduct an adaptive reperentaion learning, we propose and develop a novel alternate learning in two spaces(ALTS), basis function space and weight space, in the paper. ALTS seeks proper representations depending on the transition state from demonstration to new task and it works serially. In this way, ALTS skillfully solves the above problems via data-driven modeling and optimizing across three-phases repeatedly. To the best knowledge, it is the first to investigate LfDRL for generalization and specialty simultaneously. It is believed that this paper will not only propose a useful framework and algorithm but also stimulate further study and advance the robot's assimilation into the human lives.

This paper is organized as follows. DMPs-iLWR for policy representation and imitation learning is presented in Sect. II. Sect. III presents the policy optimization based on PI^2-DP in the scenario of robot motion planning from a perspective of stochastic optimal control. Sect. IV proposes ALTS algorithms and theoretic convergence analysis. Sect. V presents in detail the classical benchmark experiment trajectory planning via prior unknown point(s), with discussion on the experimental settings and results. Last Sect. concludes this paper with highlighs.

II. DMPs-iLWR Based Robot Motor Skill Learning

Generally, the rigid robot motion obeys a dynamic model which involves inertia matrix, Coriolis and centrifugal forces, gravity force and joint control inputs [30]. In this paper, it is assumed that underlying joint controllers can assure perfect reference tracking. So only the kinematic model is considered instead. This assumption helps us to focus on the core of problem discussed in the paper: robot movement skill acquisition and refinement.

Classical DMPs model the robot movement in each degree of freedom(DOF) as independent transformation system, which is synchronized in time dimension by a share phase variable shown as equation (1).

$$\begin{cases} \tau^2\ddot{x}_t = \underbrace{\alpha_x(\beta_x(g-x_t)-\tau\dot{x}_t)}_{spring-damping.system} + \underbrace{\Psi_\theta(s_t)s_t(g-x_0)}_{forcing.term} & transf.system \\ \tau\dot{s}_t = -\alpha_s s_t & canon.system \\ \Psi_\theta(s_t) = \frac{\sum_{i=1}^{\Gamma}\psi_i w_i}{\sum_{i=1}^{\Gamma}\psi_i} & func.approx, \end{cases} \tag{1}$$

where τ is the scaling factor for the duration of motion. x_t is the reference trajectory generated by transformation system for one DOF, s_t is the phase of the movement generated by canonical system, which decays from 1 to 0 over the same duration with transformation system. ψ_i is the Gaussian kernel function with the variance spaced equally across motion duration. w_i is associated weight. The goal g is a point attractor and x_0 is the start state. α_x,β_x,α_s are positive constants. The spring damping system denoted as a_s is modeled as a 2-order critical damping system. And forcing term presented as a_f is modeled via classical LWR methods. θ is the hyper-parameter for basis functions. Γ is the number of Gaussian kernel functions.

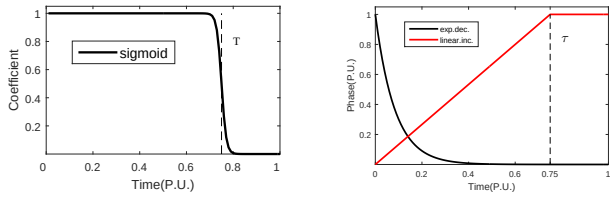
We in this paper propose ALTS which covers innovations along three phases of LfdRL. Here we begin with the DMPs-iLWR component in policy representation phase and imitation phase phase. Usually, synchronization between transform system and canonical system is considered. So both system can reach the goal (within the tolerance for exampe 2%) simultaneously. In other words, the settling time of the former is 0.4667τ when forcing term is ignored, and α_s is well assigned as $\frac{\alpha_x}{3}$ to guarantee synchronicity of the latter. Also, phase generated by canonical system in equation (1) usually decrease exponentially. It can guarantee the convergence of transform system. However such design in canonical system results in the phase being not linear correlation with time, which may cause inconvenience when statistical inference involving time is conducted. So we put forward improved Local Weighted Regression(iLWR) to solve above problem. It is shown in equation (2), which comprises a canonical system, a transformation system, and a weighted basis function. We will investigate them one by one in detail as following

$$\begin{cases} \tau^2\ddot{x}_t &= \underbrace{\alpha_x(\beta_x(g-x_t)-\dot{x}_t)}_{a_s} + \underbrace{h_t\bar{\mathbf{B}}_{s_t}\bar{\mathbf{w}}(g-x_0)}_{a_f} & \text{transf.system} \\ \tau\dot{s}_t &= \begin{cases} 1/\varrho & \text{if } s_t \le \varrho\tau \\ 0 & \text{otherwise} \end{cases} & \text{canon.system} \\ h_t &= \frac{1}{1+e^{\alpha_h(t-\varrho\tau)}} & \text{gating.system} \\ \bar{\mathbf{w}} &= [\bar{w}_1 \ \cdots \ \bar{w}_K \ \bar{w}_{K+1} \ \cdots \ \bar{w}_{2K}]^T & \text{weight} \\ \bar{\mathbf{B}}_{\mathbf{s_t}} &= [\gamma_1 s_t \ \cdots \ \gamma_K s_t \ \gamma_1 \cdots \gamma_K] & \text{basis.function,} \end{cases} \tag{2}$$

where α_h is a coefficient which regulates the steepness of gating system, ϱ is a parameter which determines the specific time when $h_t = 0.5$ and $s_t = 1$ simultaneously. Since $\frac{0.4667\tau}{0.98} = \frac{\varrho_{min}\tau}{1}$, so ϱ is in the domain of $[0.4762, 1]$ and we choose $\varrho = 0.75$ in the paper as shown in Fig.1. And γ is a component which we will depict in detail hereinafter.

Unlike our previous algorithm in DMPs for imitation learning, we adopt a slight revised canonical system similar to [4], which is described in equation (2) and τ denotes the measurement of motion duration. Usually, we choose time base τ as the per-unit value(P.U.), shown in Fig.1, in coding. Obviously, it is approximately twice the expected movement time, since spring-damping system approaches the target within 2% deviation in 0.4667τ. New canonical system equation (2) can guarantee that the phase is proportional to time in the transient process. So the distortion between $a_f(s_t)$ and $a_f(t)$ caused by canonical system in equation (1) is avoided. Hereafter we adopt terminology phase or time alternately in our proposed model.

Besides, we employ the revised gating system h_t similar to [4] to guarantee the convergence of forcing term, since



(a) h_t: the sigmoidal dacay function which mimics the step function as the gating system (b) s_t: original exponential decay phase(black) and modified linear increase phase(red)

Fig. 1. the graphic of the gating system and canonical system

it equals 0.5 at $t = \varrho\tau$ and decline to zero in exponential level after $t > \varrho\tau$ as the Fig.1(a) shows. And the fitting component modeled by iLWR results in the linear function form and is bounded because revised phase maintains 1 when it reaches. So the forcing term which is the multiplication of fitting component and gating system will converge to zero with no doubt.

Next, we will discuss the improvement of flexible adjustment. The forcing term of traditional LWR shown as

$$f(s_t) = \frac{\sum_{i=1}^{\Gamma} \psi_i(s_t) w_i}{\sum_{i=1}^{\Gamma} \psi_i(s_t)} s_t \, (g - x_0) \tag{3}$$

presents an restrictive effect of imitation learning, and

$$\psi_i = e^{\left(-\frac{1}{2\sigma_i^2}(s_t - c_i)^2\right)}$$

is the form, σ_i and c_i are appropriate variance and mean respectively. Because the default fitting trajectories must cross the origin of coordinates, which means if the phase variable is close to zero, so must be the forcing term. Therefore we revise the controlling term from pattern $(y = Ax)$ to pattern $(y = Ax + B)$ to avoid this restriction.

Now we can get the new forcing term as

$$\begin{aligned} f(s_t) &= \frac{\sum_{i=1}^{\Gamma} \psi_i(s_t) \begin{bmatrix} A_i & B_i \end{bmatrix}}{\sum_{i=1}^{\Gamma} \psi_i(s_t)} \begin{bmatrix} s_t \\ 1 \end{bmatrix} (g - x_0) \\ &= \sum_{i=1}^{\Gamma} \gamma^{(i)} (A_i s_t + B_i)(g - x_0). \end{aligned} \tag{4}$$

where

$$\gamma^{(i)} = \frac{e^{\left(-\frac{1}{2\sigma_i^2}(s_t - c_i)^2\right)}}{\sum_{j=1}^{\Gamma} e^{\left(-\frac{1}{2\sigma_j^2}(s_t - c_j)^2\right)}}, \tag{5}$$

and ψ is named as meta basis function.

Associated with the optimization policy of PI^2-DP, we can learn this two parameters(A and B) simultaneously. With this in mind, we can re-index the above components with index ι from 1 to 2Γ. Corresponding to the normal form equation (2), we can obtain

$$\bar{\mathbf{B}}_{s_t}^{(\iota)} = \begin{cases} \gamma^{(i)} s_t & \iota = i, \iota \leqslant \Gamma \\ \gamma^{(i)} & \iota = i + \Gamma, \Gamma < \iota \leqslant 2\Gamma \end{cases}$$

and accordingly the length of $\bar{\mathbf{w}}^{(\iota)}$ become 2Γ

$$\bar{\mathbf{w}}^{(\iota)} = \begin{cases} A_i & \iota = i, \iota \leqslant \Gamma \\ B_i & \iota = i + \Gamma, \Gamma < \iota \leqslant 2\Gamma \end{cases}$$

Apparently, basis function $\bar{\mathbf{B}}_{s_t}^{(\iota)}$ is compose of all meta gaussian function ψ_i. We would like to note that $\bar{\mathbf{w}}^{(\iota)}$ $(\iota \leqslant \Gamma)$ and $\bar{\mathbf{w}}^{(\iota+\Gamma)}$ $(\iota \leqslant \Gamma)$ are the slope and interception of the linear function, respectively. For the sake of simplicity, we replace $\mathbf{B}_{s_t}$ as $\mathbf{B}_t$ hereinafter.

III. Policy Improvement Based on PI^2-DP

Although DMPs-iLWR can effectively replicate and generalize robot demonstration movement, it maybe not an optimal/suboptimal policy for the task, let alone autonomously fulfill the motion different from demonstration with a high-quality level. So we combine DMPs-iLWR with a PI^2-DP policy improvement similar to [25] through stochastic optimal control to meet the requirement. Specifically, we apply Feynman-Kac theorem to derive the state value function of robot with D DOFs based on path integral, and then deduce the optimal control policy.

As we can see, the transformation system of robot with dth DOF can be expressed as

$$\begin{aligned} & \tau \ddot{x}_{d;t} = \alpha_x(\beta_x(g_d - x_{d;t}) - \dot{x}_{d;t}) + h_{d;t} \bar{\mathbf{B}}_{d;t} \mathbf{w}_d (g_d - x_0) \\ \Rightarrow \quad & \ddot{x}_{d;t} = f(x_{d;t}, \dot{x}_{d;t}) + \mathbf{B}_{d;t} \mathbf{w}_d. \end{aligned} \tag{6}$$

Next, we introduce cost $R(\xi_{d;t})$ as the cumulative return in finite horizon along the path $\xi_{d;t}$ starting in the state $\boldsymbol{\eta}_{d;t}$ at time t and ending in the state $\boldsymbol{\eta}_{d;t_N}$ at time t_N to evaluate the trajectory planning for dth DOF, where $\boldsymbol{\eta}_{d;t} = [s_t \; x_{d;t} \; \ddot{x}_{d;t}]^T$ is associated state at time t. For given robot task with D DOFs, cost $R(\xi_t)$ is shown as

$$\begin{aligned} R(\xi_t) &= \sum_{d=1}^{D} \left(\phi_{d;t_N} + \int_t^{t_N} r_{d;\zeta} d\zeta \right) \\ &= \sum_{d=1}^{D} \left[\phi_{d;t_N} + \int_t^{t_N} \left(q_{d;\zeta}^* + \frac{1}{2} \mathbf{w}_d^T Q \mathbf{w}_d \right) d\zeta \right], \end{aligned} \tag{7}$$

where t is the time index, $\phi_{d;t_N}$ is the immediate reward associated with the trajectory terminal state with dth DOF in the time index t_N, $r_{d;\zeta}$ is the immediate reward with dth DOF in the time index ζ, $q_{d;t}^*$ is a state-dependent cost function with dth DOF in the time index t, Q is an appropriate semi-define constant.

Alternatively, seeking for appropriate trajectory programming for each DOF could be treated as stochastic optimal control issue given that perturbation is applied. In other words, trajectory programming is generated by robot which consists of D stochastic dynamic sub-systems synchronized by the time. For dth DOF, it could be presented as

$$\ddot{x}_{d;t} = f_{d;t} + \mathbf{B}_{d;t} \left(\mathbf{w}_d + \boldsymbol{\varepsilon}_{d;t} \right), \tag{8}$$

where ε_d is a random variable with zero expectation and

variance as Λ_d. And associated HJB equation is

$$-\partial_{d;t}V_t = \min_{\mathbf{w}_d}\Big(r_{d;t} + \left(\nabla_{\dot{x}_{d;t}}V_t\right)^T f_{d;t} + \frac{1}{2}trace\left(\left(\nabla_{\dot{x}_{d;t}\dot{x}_{d;t}}V_t\right)\right)\mathbf{B}_{d;t}\Lambda_{d;t}\mathbf{B}_{d;t}^T\Big), \quad (9)$$

where the value function under the optimal parameterized control policy is V_t. It can be expressed as

$$V_t = \min_{\mathbf{w}} \mathbb{E}\left[R\left(\xi_t\right)\right] \quad (10)$$

and $\mathbf{w} = \{\mathbf{w}_d\}_1^{\mathrm{D}}$ is a vector with component $\mathbf{w}_d$.

Substituting value function with $\Psi_t = e^{-\frac{V_t}{\lambda}}$, we can obtain

$$-\partial_{d;t}\Psi_t = -\frac{1}{\lambda}r_{d;t}\Psi_t + \left(\nabla_{\dot{x}_{d;t}}\Psi_t\right)^T f_{d;t} + \frac{1}{2}\mathrm{trace}\left(\left(\nabla_{\dot{x}_{d;t}\dot{x}_{d;t}}\Psi_t\right)\right)\mathbf{B}_{d;t}\Lambda_{d;t}\mathbf{B}_{d;t}^T, \quad (11)$$

which is a standard backward Kolmogorov partial differential equation. Then we apply Feynman-Kac formula and gain

$$\Psi_t = \sum_{d=1}^{D}\left\{\int_t^{t_N} p\left(\xi_{\cdot;t}\left|\boldsymbol{\eta}_{\cdot;t}\right.\right) \exp\left[-\frac{1}{\lambda}\left(\phi_{d;t_N} + \int_t^{t_N} r_{d;y}dy\right)\right] d\left(\delta\xi_{d;t}\right)\right\}, \quad (12)$$

where $\delta\xi_{d;t}$ is all the variational paths with dth DOF, and $d\left(\delta\xi_{d;t}\right)$ is all infinitesimal differential along the path with dth DOF, $p\left(\xi_{\cdot;t}\left|\boldsymbol{\eta}_{\cdot;t}\right.\right)$ is probability of sample path conditioned on the start state $\boldsymbol{\eta}_{\cdot;t}$ with all DOFs, $\cdot$ indicate all DOFs from $d=1$ to D and λ is a constant. Without losing generality, we can discretize it with sufficient small time interval. Thus the above formula is transformed as

$$\Psi_i = \sum_{d=1}^{D}\left\{\lim_{\Delta t\to 0}\int p\left(\xi_{\cdot;i}\left|\boldsymbol{\eta}_{\cdot;i}\right.\right) \cdot\exp\left[-\frac{1}{\lambda}\left(\phi_{d;N} + \sum_{j=i}^{N-1} r_{d;j}\right)\right] d\xi_{d;i}\right\}, \quad (13)$$

where $\xi_{d;i} = \left(\boldsymbol{\eta}_{d;i}, \boldsymbol{\eta}_{d;i+1}, \cdots, \boldsymbol{\eta}_{d;N}\right)$ is a sample trajectory with dth DOF, i is an abbreviation for time index t_i.

In practical algorithm, strategy of per-basis exploration noise is applied. In other words, for specific s_t, iLWR-PI^2 seeks the maxim $\gamma^{(i)}$ in equation (5) for coefficient index i, then perturbs $\mathbf{w}^{(i)} + \varepsilon$ (i.e. $A_i + \varepsilon$) as well as $\mathbf{w}^{(i+\Gamma)} + \varepsilon$ (i.e. $B_i + \varepsilon$) simultaneously. That is PI^2 with dual perturbation denoted as PI^2-DP.

As we known, $p\left(\xi_{d;i}\left|\boldsymbol{\eta}_{d;i}\right.\right)$ is the probability of sample path conditioned on the start state $\boldsymbol{\eta}_{d;t_i}$. Specifically, it is expressed as

$$\begin{aligned} p\left(\xi_{d;i}\left|\boldsymbol{\eta}_{d;i}\right.\right) &= p\left(\boldsymbol{\eta}_{d;N}, \cdots, \boldsymbol{\eta}_{d;i+1}\left|\boldsymbol{\eta}_{d;i}\right.\right) \\ &= \Pi_{j=i}^{N-1} p\left(\boldsymbol{\eta}_{d;j+1}\left|\boldsymbol{\eta}_{d;j}\right.\right) \\ &\propto \exp\left(-\tfrac{1}{2\lambda}\sum_{j=i}^{N-1}\left\|\mathbf{B}_{d;j}\left(\mathbf{w}_{d;j} + \boldsymbol{\varepsilon}_{d;j}\right)\right\|^2_{\mathbf{H}_{d;j}^{-1}}\right) \end{aligned}$$

on the consideration of $\boldsymbol{\varepsilon}$ is a Gaussian distribution, where $\mathbf{H}_{d;j} = \mathbf{B}_{d;j}Q^{-1}\mathbf{B}_{d;j}^T$.

So we can gain

$$p\left(\xi_{\cdot;t}\left|\boldsymbol{\eta}_{\cdot;t}\right.\right) = \prod_{d=1}^{D} p\left(\xi_{d;t}\left|\boldsymbol{\eta}_{d;t}\right.\right)$$

Substituting it into equation (13), we can derive the formula as

$$\Psi_i \propto \sum_{d=1}^{D}\left(\lim_{\Delta t\to 0}\int \exp\left(-\frac{1}{\lambda}S\left(\xi_{d;i}\right)\right) d\xi_{d;i}\right) \quad (14)$$

and

$$S\left(\xi_{d;i}\right) = \phi_{d;N} + \sum_{j=i}^{N-1} q_{d;j} + \frac{1}{2}\sum_{j=i}^{N-1}\left(\mathbf{w}_d + \boldsymbol{\varepsilon}_{d;j}\right)^T \cdot\frac{\mathbf{B}_{d;j}\mathbf{B}_{d;j}^T}{\mathbf{B}_{d;j}^T R^{-1}\mathbf{B}_{d;j}}\left(\mathbf{w}_d + \boldsymbol{\varepsilon}_{d;j}\right) + \frac{\lambda}{2}\sum_{j=i}^{N-1}\ln\left|\mathbf{H}_{d;j}\right|. \quad (15)$$

We would like to point out that $\frac{\lambda}{2}\sum_{j=i}^{N-1}\ln\left|\mathbf{H}_{d;j}\right|$ in original PI^2 is removed given that basis function is fixed. However, this term is indispensable in our algorithms because of varying basis function in the procedure of ALTS.

Based on the preceding deduction, we can obtain the optimal time-variant policy with value function V_i as

$$\begin{aligned} \mathbf{w}_{d;i} &= -Q^{-1}\mathbf{B}_{d;j}^T\left(\nabla_{\dot{x}_{d;i}}V_i\right) \\ &= \textstyle\int P\left(\xi_i\right) u\left(\xi_{d;i}\right) d\xi_i, \end{aligned} \quad (16)$$

where

$$P\left(\xi_i\right) = \frac{e^{-\frac{1}{\lambda}\left(\sum_{d=1}^{D} S\left(\xi_{d;i}\right)\right)}}{\int e^{-\frac{1}{\lambda}\left(\sum_{d=1}^{D} S\left(\xi_{d;i}\right)\right)} d\xi_i} \quad (17)$$

$$u\left(\xi_{d;i}\right) = \frac{R^{-1}\mathbf{B}_{d;i}\mathbf{B}_{d;i}^T}{\mathbf{B}_{d;i}^T Q^{-1}\mathbf{B}_{d;i}}\left(\mathbf{w}_d + \boldsymbol{\varepsilon}_{d;i}\right) \quad (18)$$

Specifically, $P\left(\xi_i\right)$ is the path depended probability distribution with robot, and $u\left(\xi_{d;i}\right)$ is respective local optimal control derived by value function for dth DOF.

We would like to point out that it is unrealistic to integral along all the variational paths under the theoretical requirement. In practical engineering, we usually carry out K roll-outs. And for specific time index i, we gain $P\left(\xi_{\cdot;i;k}\right)$ as probability of robot (all DOFs) based on kth episodic sample in time index i which is similar to softmax function represented as

$$P\left(\xi_{\cdot;i;k}\right) = \frac{e^{-\frac{1}{\lambda}\left(\sum_{d=1}^{D} S\left(\xi_{d;i;k}\right)\right)}}{\sum_{k=1}^{K} e^{-\frac{1}{\lambda}\left(\sum_{d=1}^{D} S\left(\xi_{d;i;k}\right)\right)}}, \quad (19)$$

where k is the index of K episodes, $\xi_{d;i;k}$ is trajectory sample with dth DOF in kth episode in time index i.

Apparently, it is $P\left(\xi_{\cdot;i;k}\right)$ that guides and coordinates all DOFs' local $u\left(\xi_{d;i}\right)$ to drive the robot to approach an optimal/sub-optimal point as a whole. As an result, the respective weight $\mathbf{w}_{d,i}$ with dth DOF in time i is obtained. Since weight is usually a constant instead of a variable, we achieve weighted average of adjustment $\Delta\mathbf{w}_{d;i} = \mathbf{w}_{d;i} - \mathbf{w}^{old}$ across N time index as equivalent time-invariant policy, which could

be expressed as

$$\Delta \mathbf{w}_d = \frac{\sum_{i=0}^{N-1} (N-i)\, \mathbf{B}_{d;i} \Delta \mathbf{w}_{d;i}}{\sum_{i=0}^{N-1} (N-i)\, \mathbf{B}_{d;i}}. \tag{20}$$

IV. Basis Function Reconstruction and alternate learning in two spaces

The embodiment feature specifies the robot intelligence is strongly linked to the robot morphology and its environmental interaction capability [31]. So the basis function in the policy representation is the key to establish the intelligence feasibility based on the robot kinematics. If the robot needs to finish a different task from the demonstration, the current basis function based on the demonstrated task will not be appropriate for the new task.

The question arises as to how to obtain the suitable basis function for new task on conditioan that demonstration data are avaliable at the start. Motivated by the gradual and iterative process of cognition and practice for human's new skill acquisition [32], [33], we assume that robot skill acquisition also has corresponding processes. Cognition process is performed as searching appropriate latent pattern based on good experience so far by means of representation learning, practice process is performed as applying acquired new pattern to explore better trajectory or $R(\xi)$ by means of reinforcing learning. Two processes are carried out alternately.

Based on the above idea and analysis, we put forward DMPs-iLWR together the alternate learning in two spaces(ALTS) to conduct motion skill transfer from demonstration to new task. For the sake of simplicity, we denote it as iLWR-PI^2-AL. It's workflow is described as following.

Policy representation (iLWR) for motor skill is firstly developed in reference to maximum entropy. It means alike meta Gaussian functions ψ_i with identical variances are equally allocated during the phase, so that it can enable the policy to learn the new motions with the fixed number of basis function. Afterwards, LfD is performed to find suitable weights for reproduce the demonstration. Then, RL(PI^2-DP) is employed to find right weight until the cost doesn't reduce significantly anymore. It indicates that the algorithm has already found optimal/sub-optimal approximation point on the corresponding weight space for the new task with the current basis function. Thus DB-Kmeans is applied to cluster on the data generated by those candidate elites represented as $\{s_t, a_f/\lceil \tau^2(g-x_0)\rceil\}$. Then EM-GMM is adopted to obtain $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ of every multivariate Gaussian components. Next, s_t-axis component of $\boldsymbol{\mu}_i$ is taken and assigned to μ_i. $\boldsymbol{\Sigma}_i$ is decomposed into eigenvalue and eigenvector which is orthonormal. Let's denote $\boldsymbol{\Upsilon}_i$ as eigenvector which is with respective to the orientation of s_t-axis and ν_i as correspond eigenvalue. So the norm of vector which is the projection of $\sqrt{\nu_i}\boldsymbol{\Upsilon}_i$ on s_t-axis is assigned to σ_i. In this way, appropriate parameters μ_i and σ_i for respective meta basis function ψ_i are achieved. Finally, these parameters will be assign to $\gamma^{(i)} s_t$ and $\gamma^{(i)}$ to reconstruct the new basis function $\bar{\mathbf{B}}_{s_t}$. In a sense, more appropriate basis functions are constructed adaptively based on data-driven according to the character of targeted task. Also, LfD is performed to seek weights to replicate the best trajectories so far with a posterior maximization. Based on them, we again apply RL(PI^2-DP) to search the best approximation in the new space. This procedure repeat until a satisfied trajectory is obtained. Detail of ALTS is depicted in the form of pseudo codes. Please see algorithms 1 hereafter and algorithms 2-4 in appendix.

Algorithm 1 ALTS

Require: N: maximum elapsed steps for each episode
D: number of joints
MaxIter: maximum iteration numbers
Γ: number of meta basis functions
K: maximum number of rollouts
Ensure: trajectory ξ, reward $R(\xi)$

```
Initialize iteration index p = 0
Initialize basis function B ← {B_d}_1^D according to maximum entropy of meta basis function
ξ ← {x_j, u_j}_1^N // trajectory including state,action from demonstration
{ξ_d}_1^D ← ξ // resolved into each DOF
for d = 1 to D do
    w_d ←iLWR_LfD(ξ_d, B_d) // conduct imitation learning
end for
w ← {w_d}_1^D
while p < MaxIter do
    p = p + 1
    w^new, {ξ^k, R(ξ^k)}_1^K ← Weight_Learning(w, B)
    ξ, R(ξ) ← Policy_Evaluation(w^new, B)
    if Ṙ(ξ_N) < 0 and |Ṙ(ξ_N)| < Thre for 3 times in a row then
        {ξ*} ←sel— {ξ^k, R(ξ^k)}_1^K // select elite set
        {ξ*_d}_1^D ← {ξ*} // resolved into each DOF
        for d = 1 to D do
            w_d^new, B_d^new ←Basis Learning(ξ*_d, Γ)
        end for
    end if
    w, B ← w^new, B^new
end while
ξ, R(ξ) ← Policy_Evaluation(w, B)
```

We now proceed to investigate the theoretic analysis of ALTS algorithm. We use $\mathbf{s} \in \mathcal{S}$ to denote states, $\mathbf{a} \in \mathcal{A}$ to denote actions, $r(\mathbf{s}_t, \mathbf{a}_t)$ to denote reward function. State $\mathbf{s}_{t+1}$ evolves from $\{\mathbf{s}_t, \mathbf{a}_t\}$ in the scenario of discrete-time finite horizon. The problem aims to find a policy parameter $\mathbf{w}$ that minimize the total reward $\Sigma_t r(\mathbf{s}_t, \mathbf{a}_t)$ on episode, no matter state-index policy $p(\mathbf{a}_t|\mathbf{s}_t, \mathbf{w})$ or time-index policy $p(\mathbf{a}_t|t, \mathbf{w})$ is applied. The expectation is taken under the policy's trajectory distribution $p(\xi)$, presented by

$$p(\xi) = p(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_N, \mathbf{a}_N).$$

Here, we investigate RL formulation from a respective which is illustrated by a HMM-like graphic model with optimality variables $\mathbf{o}_t$ shown in Fig.2. $\mathbf{o}_t$ is a binary random, where $\mathbf{o}_t = 1$ denotes that time step t is optimal. No loss of

generality, we can choose the distribution over this variable as

$$p\left(\mathbf{o}_t = 1|\mathbf{s}_t, \mathbf{a}_t\right) = \exp\left(-r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right).$$

Thus we can obtain

$$p\left(\mathbf{o}_{1:N} = 1|\xi\right) = \exp\left(\sum_{t=1}^{N} -r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right),$$

which can be interpreted as the event of obtain minimum total reward by choosing a serial optimal actions.

gragic model.pdf

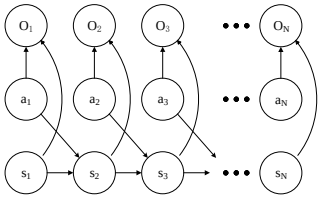


Fig. 2. A diagram of graphical model

Let's consider optimality variables $\mathbf{o}_{1:N}$ as observable data, trajectory ξ as hidden state. Let weight $\mathbf{w}$ present model parameters controlling distribution of $\mathbf{o}_{1:N}$ and ξ. Let $q_z\left(\xi\,|\mathbf{o}_{1:N}\right)$ be a family of distribution of ξ parameterized by latent z which is the parameters of meta basis function such as $\{c_j, \sigma_j\}_1^\Gamma$. It can be interpreted as the fact that the distribution of ξ is determined by basis $\mathbf{B}$ via z on condition that event of optimal actions occurred.

What we search is the appropriate policy to fulfill the task with minimum total reward. It can be presented as

$$\arg\max_{\mathbf{w}} \mathbb{E}[p(\mathbf{w}|\mathbf{o}_{1:N})] = \arg\max_{\mathbf{w}} \mathbb{E}[\log p(\mathbf{w}, \mathbf{o}_{1:N})]. \tag{21}$$

Note

$$\begin{aligned}
&\mathbb{E}[\log p(\mathbf{w}, \mathbf{o}_{1:N})] \\
=&\sum_{\xi} q_z\left(\xi\,|\mathbf{o}_{1:N}\right) \log p(\mathbf{w}, \mathbf{o}_{1:N}) &(22)\\
=&\sum_{\xi} q_z\left(\xi\,|\mathbf{o}_{1:N}\right) \log \frac{p(\mathbf{w}, \xi, \mathbf{o}_{1:N})}{q_z\left(\xi\,|\mathbf{o}_{1:N}\right)} \frac{q_z\left(\xi\,|\mathbf{o}_{1:N}\right)}{p\left(\xi|\mathbf{w}, \mathbf{o}_{1:N}\right)} &(23)\\
=&\mathcal{L}\left(z, \mathbf{w}\right) + \mathcal{K}\left(z, \mathbf{w}\right) &(24)
\end{aligned}$$

where

$$\mathcal{L}\left(z, \mathbf{w}\right) = \sum_{\xi} q_z\left(\xi\,|\mathbf{o}_{1:N}\right) \log \frac{p(\mathbf{w}, \xi, \mathbf{o}_{1:N})}{q_z\left(\xi\,|\mathbf{o}_{1:N}\right)} \tag{25}$$

$$\mathcal{K}\left(z, \mathbf{w}\right) = \sum_{\xi} q_z\left(\xi\,|\mathbf{o}_{1:N}\right) \log \frac{q_z\left(\xi\,|\mathbf{o}_{1:N}\right)}{p\left(\xi|\mathbf{w}, \mathbf{o}_{1:N}\right)} \tag{26}$$

We would like to point out that $\mathcal{K}\left(z, \mathbf{w}\right)$ is the KL divergence $\mathrm{KL}\left(q_z\left(\xi\,|\mathbf{o}_{1:N}\right)||p\left(\xi|\mathbf{w}, \mathbf{o}_{1:N}\right)\right)$. Since KL divergence is non-negative, so $\mathcal{L}\left(z, \mathbf{w}\right)$ is a lower bound on $\mathbb{E}[\log p(\mathbf{w}, \mathbf{o}_{1:N})]$. ALTS works like a generalized expectation maximization to improve this bound iteratively to approach the maximum. In other words, we can gain a sequence of two-tuples $\left(\mathbf{w}^{(1)}, z^{(1)}\right)$, $\left(\mathbf{w}^{(2)}, z^{(2)}\right)$, $\cdots$ by iteration in two spaces. Specifically, in the procedure of $k+1$ iteration, there are E Step and M Step.

E Step: In this step, ALTS manage to search $z^{(k+1)}$ to satisfy the equation as

$$z^{(k+1)} = \arg\max_{z} \mathcal{L}\left(z, \mathbf{w}^{(k)}\right). \tag{27}$$

Since $\mathcal{L}\left(z^{(k)}, \mathbf{w}^{(k)}\right) = \mathbb{E}\left[\log p\left(\mathbf{w}^{(k)}, \mathbf{o}_{1:N}\right)\right] - \mathcal{K}\left(z^{(k)}, \mathbf{w}^{(k)}\right)$ and $\mathbb{E}\left[\log p\left(\mathbf{w}^{(k)}, \mathbf{o}_{1:N}\right)\right]$ is a constant, this step amounts to minimizing $\mathcal{K}\left(z, \mathbf{w}^{(k)}\right)$ with respect to z. That is, it chooses an appropriate z to construct a member of the variational family q_z which is as close as possible to the current $p\left(\xi|\mathbf{w}, \mathbf{o}_{1:N}\right)$. From the perspective of algorithm implementation, it involves clustering by DB-Kmeans, searching z by EM-GMM and identifying new basis $\mathbf{B}$ by iLWR. We would like to point out that basis functions' adaptive evolving is definitely corresponding to the learning in the basis function space.

M Step: In this step , ALTS manage to search $\mathbf{w}^{(k+1)}$ to satisfy the equation as

$$\mathbf{w}^{(k+1)} = \arg\max_{\mathbf{w}} \mathcal{L}\left(z^{(k+1)}, \mathbf{w}\right). \tag{28}$$

From graphical model illustrated in Fig.2, we can get

$$\begin{aligned}
p(\xi, \mathbf{o}_{1:N}) =& p(\mathbf{s}_1)\prod_{t=1}^{N} p(\boldsymbol{o}_t = 1|\mathbf{s}_t, \mathbf{a}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \\
=& p(\mathbf{s}_1) p(\mathbf{o}_{1:N} = 1|\xi) \prod_{t=1}^{N} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \\
=& \vartheta \cdot \exp\left(\sum_{t=1}^{N} -r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right).
\end{aligned} \tag{29}$$

where $\vartheta = p\left(\mathbf{s}_1\right)\prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. For deterministic dynamics, ϑ is a constant for all trajectories that are dynamic feasible. So $p\left(\xi, \mathbf{o}_{1:N}\right) \propto \exp\left(\sum_{t=1}^{N} -r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)$ is derived. And $p(\mathbf{w}|\xi, \mathbf{o}_{1:N})$ is the probability of the event $\mathbf{w} = \mathbf{w}^{opt}$ when variational distribution ξ is fixed and actions is optimal. When $\exp\left(\sum_{t=1}^{N} -r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)$ is minimized, $p\left(\mathbf{w}|\xi, \mathbf{o}_{1:N}\right)$ equals 1 at its maximum value. Considering $p(\mathbf{w}, \xi, \mathbf{o}_{1:N}) = p\left(\xi, \mathbf{o}_{1:N}\right) p(\mathbf{w}|\xi, \mathbf{o}_{1:N})$, so maximizing $\mathcal{L}\left(z^{(k+1)}, \mathbf{w}\right)$ with respect to $\mathbf{w}$ is equivalent to maximizing total reward on episode since q_z is fixed. From the perspective of algorithm implementation, it involves variational searching by PI^2-DP. We would like to point out that reward-guided searching is definitely corresponding to the learning in weight space.

Briefly, we can obtain $\mathcal{L}\left(z^{(k+1)}, \mathbf{w}^{(k)}\right) \geqslant \mathcal{L}\left(z^{(k)}, \mathbf{w}^{(k)}\right)$ in E step and $\mathcal{L}\left(z^{(k+1)}, \mathbf{w}^{(k+1)}\right) \geqslant \mathcal{L}\left(z^{(k+1)}, \mathbf{w}^{(k)}\right)$ in M step. In this way, the monotone convergence of ALTS is guaranteed, and a sequence of two-tuples is constructed until a local minimum value is reached, so that the demonstration task and new one are joined.

V. Simulation Experiment

A. Application Scenario I: SCARA

In this section, we employ the SCARA manipulator shown in [34] to perform motor skill acquisition experiments. The SCARA manipulator has four joints: revolute joints q_1, q_2, q_3

and prismatic joint q_4. l_1 denotes the length of the link 1, and l_2 denotes the length of the link 2.

Skill acquisition experiences are described as below. First, man gives a demonstration of one stroke motion with which the end-effector of the robot moves from the initial point to end point with duration is $5s$. Then a small box is arbitrarily located in the space, where the robot can reach (excluding the initial and end points). Therefore the robot arm should be able to move and pass this via-point (*e.g.* sucking, photographing or laser marking) from the initial point to end point in the same time. Moreover, the cost of equation (31) is satisfied.

Assumed there is a suction force applied to pick up the target block, so we don't pay attention to the orientation of the joint. Besides, prismatic joint is perpendicular to others three revolution joints. Also, we suppose that the velocity of it is much quicker that of other three evolute joints. Based on the above assumptions, we omit the joints $q3$ and $q4$, then deduce the forward kinematic expression for simplicity shown

$$\begin{cases} \tilde{x} &= -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) \\ \tilde{y} &= l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ \tilde{z} &= l_0 \end{cases} \tag{30}$$

It is a classical via-point benchmark in the X-Y cross-section plane with $\tilde{z} = l_0$. As for the SCARA robot in the experiment, $l_0 = l_1 = l_2 = 10cm$ are assumed. Obviously the start point is $(0, 10, 10)$ in base frame(operation space).

B. Experiment I-1:Trajectory programming via a given point

We set the terminal point with $(-16, 4.5, 10)cm$ in base frame which is corresponding to $(0.7068, 1.17964, 0)rad$ (unit will be omitted later) and also expect manipulator to grasp at the specific position $(-6, 18, 10)cm$ in Cartesian coordinate . Here we adopt joint trajectory as $(0, 0, 0) \rightarrow (-0.000, 0.6435, 0) \rightarrow (0.7068, 1.17964, 0)$ based on a traditional min-energy criterion in engineering.

Here iLWR-PI^2-AL will be evaluated via different experiments. Table I shows the results. The cost function J is defined as

$$\sum_{d=1}^{D} \left(0.5 \sum_{i=1}^{N-1} (10^3 \ddot{x}_{d;i}^2 + a_{d;i;f}^2) + 10^{10} (x_{d;m} - x_{d;m;v})^2 + 10^3 \left[\dot{x}_{d;N}^2 + (x_{d;N} - x_{d;N;g})^2 \right] + 0.1 \sum_{i=1}^{N-1} \mathbf{B}_{d;i} \mathbf{B}_{d;i}^T \right) \tag{31}$$

where i is the time index from 1 to N, d is the joint index from 1 to D. D equals 3 here. Besides, $x_{d;N;g}$ denotes the expected position of point d when the task ends. When the time index equals m, $x_{d;m}$ is the position of joint d which is corresponding to the expected via-point $x_{d;m;v}$. Apparently, $\ddot{x}_{d;i}$ and $a_{d;i;f}$ are acceleration and forcing term relevant to joint d at the time index i, respectively. $\dot{x}_{d;N}$ is the velocity of joint d when time index is N. $\mathbf{B}_{d;i}$ is the control matrix in equation (8) with joint d when time index is i.

As for Table I, we evaluate the respective total cost when we vary the time index from $1.25s$ to $1.95s$ with expected via-point $(-0.000, 0.6435, 0)$ fixed in joint spaces within the duration is $5s$. Each row of data was average total cost from 30 experiments. The results demonstrate that the proposed iLWR-PI^2-AL methods can achieve better performance than original LWR-PI^2 and iLWR-PI^2. Compared with the improvemnt of iLWR, ALTS seems to make more contributions.

TABLE I
THE FINAL COSTS WHEN ALGORITHM STOPS

Time	Cost$_1$(LWR-PI^2)	Cost$_2$(iLWR-PI^2)	Cost$_3$(iLWR-PI^2-AL)
1.25	6.23E+07	6.07E+07	3.23E+07
1.30	5.74E+07	5.72E+07	3.15E+07
1.35	5.58E+07	5.28E+07	3.12E+07
1.40	5.55E+07	5.16E+07	3.07E+07
1.45	5.62E+07	4.52E+07	3.03E+07
1.50	5.89E+07	4.91E+07	3.05E+07
1.55	5.08E+07	4.82E+07	3.13E+07
1.60	6.57E+07	4.86E+07	3.13E+07
1.65	6.25E+07	5.44E+07	3.24E+07
1.70	7.63E+07	5.06E+07	3.37E+07
1.75	8.55E+07	5.13E+07	3.51E+07
1.80	9.13E+07	5.47E+07	3.54E+07
1.85	1.25E+08	5.41E+07	3.73E+07
1.90	1.40E+08	6.71E+07	3.92E+07
1.95	1.52E+08	6.89E+07	4.06E+07
Mean	8.37E+07	5.43E+07	3.35E+07

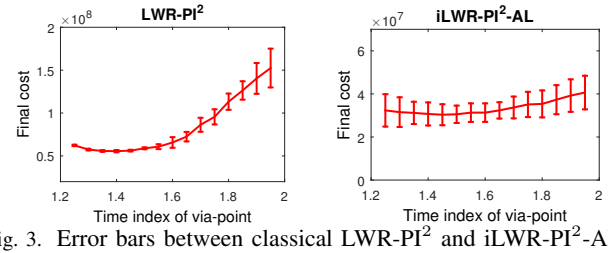


Fig. 3. Error bars between classical LWR-PI^2 and iLWR-PI^2-AL

To be clearly, In Fig.3 we plot two error bars to demonstrate cost change when the time index for via-point varies. Besides, we compare the learning rate between two methods. The total cost vary along with iteration number are shown in Fig.4(left) and Fig.4(right), respectively. Quicker coverage speed of iLWR-PI^2-AL is manifest.

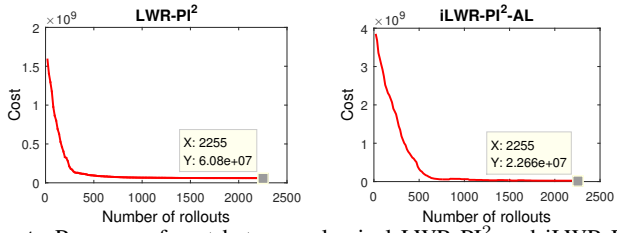


Fig. 4. Progress of cost between classical LWR-PI^2 and iLWR-PI^2-AL

And Fig.5 shows the typical amelioration of joint trajectories (3 joints in joint space) from relevant demonstration to targeted improvement. The green lines in both figures indicate the joint's trajectories planning learned by LfD. The red lines on the right figure represent the final trajectories learned by iLWR-PI^2-AL, which pass by the $(-8.845e-005, 0.6434, 0)$ at time index equals $0.2s$. Meanwhile, the red lines on the left figure represent the final trajectories learned by LWR-PI^2, which pass by the $(1.307e-004, 0.6437, 0)$ at the same time index.

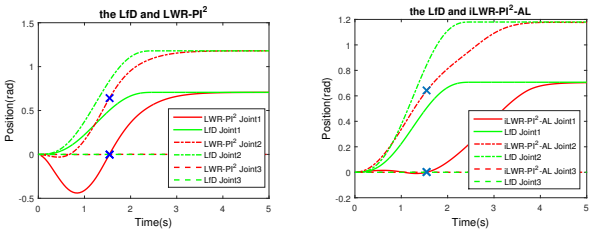


Fig. 5. Comparison of joint's trajectories between classical LWR-PI^2 and iLWR-PI^2-AL

C. Experiment I-2:Trajectory programming via two given points

We now proceed to exhibit ALTS's learning ability reinforced by the adaptive evolving of meta basis functions. We carefully design the experiments as adding another different via-point in the trajectory programming. Both via-points are with distinct distance in space whereas in very small time interval. In other words, their time index fall on the domain of the same basis function.

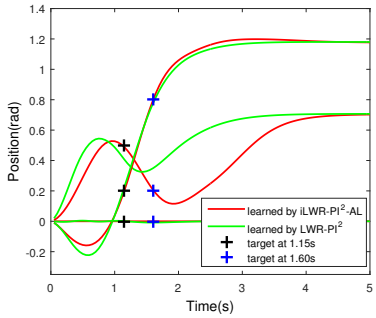
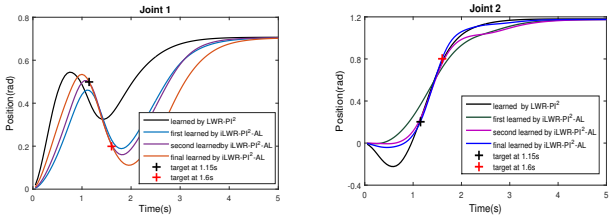


Fig. 6. Survey of the motor skill acquisition(via two given points)

As depicted in LWR-PI^2, its basis functions are evenly distributed across the phase dimension. Usually, it is impossible to only adjust weights to make a sudden curve change within the domain of identical basis function. So it is unfeasible to performance via-two-points task successfully unless the distribution of the basis functions can be changed adaptively to meet the task's requirements. In this experiment, the joints are demand to pass through joints point $(0.5, 0.2, 0)$ at $1.15s$ and $(0.2, 0.8, 0)$ at $1.60s$, which is labeled as '+' in Fig.6.

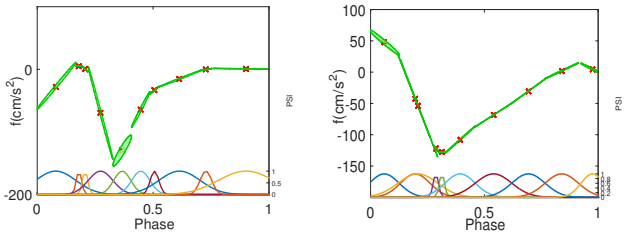
The learning progress of a typical ALTS running are displayed in Fig.7. In order to show it clearly, we only display the first, second and final learned by iLWR-PI^2-AL and LWR-PI^2, respectively.

We all know that the meta basis function ψ is evenly distributed before iLWR-PI^2 learning. In Fig.8(a), as we can see, the shape and scatter of meta basis function generated from imitation learning (the first reconstruction of meta basis function) sounds arbitrary to some degree. However, after several iteration of ALTS, the shape and scatter of meta basis function have made a targeted adjustment and improvement in Fig.8(b). The kernel nearby $1.15s$ (phase=0.3067) and $1.60s$ (phase=0.4267) evolving into two distinct meta basis functions. Each of them is with a very compact projection in the time dimension, and with significant different projection in the acceleration dimension. This kind of construction and distribution of associated meta basis functions are very suitable for the above target task (via specific points at $1.15s$ and $1.60s$). It can be regarded as reflection of embodiment intelligence.



(a) Progress of the motor skill acquisition of joint1 (via two given points) (b) Progress of the motor skill acquisition of joint2 (via two given points)

Fig. 7. Progress of the motor skill acquisition



(a) The distribution of meta basis functions by using iLWR-PI^2-AL under first recombination (b) The distribution of meta basis functions by using iLMR-PI^2-AL under final recombination

Fig. 8. Adaptive evolving of meta basis functions I

D. Application Scenario II: 10 DOF planar

In this section, we will use a 10 DOF planar to inspect iLWR-PI^2-AL algorithm. The 10 DOF planar has 10 revolute joints, and the length of each link is 10cm. This robot looks like a multi-segment snake in a plane. According to this structure, we can calculate the end-effector coordinates by using angle of joints$(q_1, q_2, \ldots, q_{10})$ as shown in equation (32)

$$\begin{cases} \tilde{x} &= lCq_1 + lC(q_1+q_2) + \cdots + lC(q_1+\ldots+q_{10}) \\ \tilde{y} &= lSq_1 + lS(q_1+q_2) + \ldots + lS(q_1+\cdots+q_{10}) \\ \tilde{z} &= 0 \end{cases} \tag{32}$$

In equation (32), l denotes the length of each link, C is abbreviation of cos, and S is abbreviation of sin. $q_1, q_2, q_3, \ldots, q_{10}$ denote the current angle of each joint. $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$ denote the position of end-effector, $\tilde{z}$ equals zero because the 10 DOF planar can only move in a 2D coordinate system.

E. Experiment II-1:Trajectory programming via a given point

In this experiment, we set the start point as $(0, 0, \cdots, 0)_{10\times1}$ * in joint space and terminal point $(0.5, 0.5, \cdots, 0.5)_{10\times1}$ as the end point in joint space. Also we set $(0.3, 0.2, 0.4, 0.2, 0.1, 0.2, 0.2, 0.2, 0.2, 0.2)$ as the via

*All joint radians of the 10 DOF planar are zero. The value inside bracket indicates respective joint radian, and subscript indicates the dimension.

point and the cost function is in the form similar to equation (31) with D equals 10.

As shown in Table II, we have investigate the difference among LWR-PI^2, iLWR-PI^2 and iLWR-PI^2-AL, further evaluate the respective cost when we vary the time index from $1.25s$ to $1.95s$. In Fig.9, we also plot two error bars to describe the process of change for the cost on condition that time index for via-point varying from $1.25s$ to $1.95s$.

TABLE II
THE FINAL COSTS WHEN ALGORITHM STOPS

Time	Cost$_1$(LWR-PI^2)	Cost$_2$(iLWR-PI^2)	Cost$_3$(iLWR-PI^2-AL)
1.25	2.42E+07	1.32E+07	1.14E+07
1.30	2.78E+07	1.38E+07	1.37E+07
1.35	3.35E+07	1.30E+07	1.26E+07
1.40	3.72E+07	1.30E+07	1.05E+07
1.45	4.31E+07	1.40E+07	1.15E+07
1.50	4.86E+07	1.41E+07	1.27E+07
1.55	5.42E+07	1.39E+07	1.25E+07
1.60	6.62E+07	1.34E+07	1.25E+07
1.65	7.90E+07	1.37E+07	1.03E+07
1.70	1.05E+08	1.35E+07	1.07E+07
1.75	1.68E+08	1.37E+07	1.05E+07
1.80	2.18E+08	1.29E+07	1.22E+07
1.85	3.15E+08	1.31E+07	1.20E+07
1.90	3.94E+08	1.41E+07	1.11E+07
1.95	4.94E+08	1.41E+07	1.29E+07
Mean	1.40E+08	1.35E+07	1.18E+07

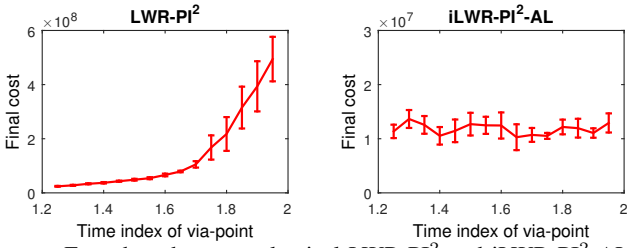


Fig. 9. Error bars between classical LWR-PI^2 and iLWR-PI^2-AL

According to this table we can easily know the cost of iLWR-PI^2-AL is smaller than that of LWR-PI^2. Besides, experiment illustrates that cost of iLWR-PI^2-AL drops faster than that of LWR-PI^2 shown in Fig.10.

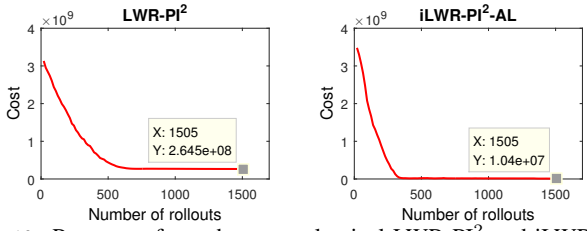


Fig. 10. Progress of cost between classical LWR-PI^2 and iLWR-PI^2-AL

The real trajectory is shown in In Fig.11. The left figure is learned by LWR-PI^2 and the right figure is learned by iLWR-PI^2-AL. They should pass through point $(2.541, 8.224)cm$ (unit will be omitted later) at time index $1.0s$. The trajectory pass through point $(2.777, 8.185)$ at time index $1.0s$ in the left of Fig.11 and the point $(2.541, 8.217)$ at time index $1.0s$ in the right of Fig.11, respectively.

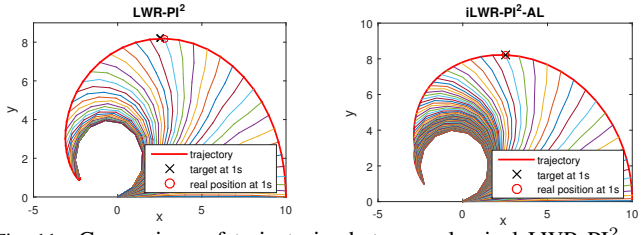


Fig. 11. Comparison of trajectories between classical LWR-PI^2 and iLWR-PI^2-AL

F. Experiment II-2:Trajectory programming via two given points

In the last experiments, we have driven 10 DOF planar via a certain point at a certain time index. At this part, we add another different via-point in the trajectory programming as we have done in SCARA. The trajectory should via point $(2.541, 8.224)$ at 1.15s and via point $(-0.5281, 6.654)$ at 1.60s which are labeled as '$\times$' in Fig.12 .

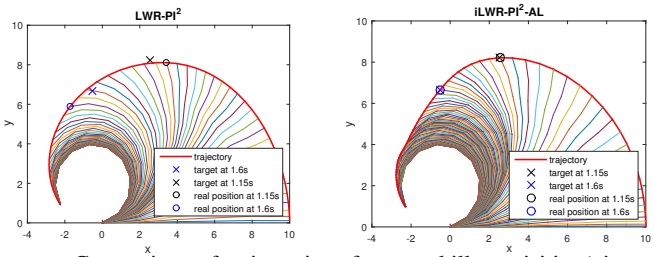


Fig. 12. Comparison of trajectories of motor skill acquisition(via two given points) between classical LWR-PI^2 and iLWR-PI^2-AL

As described in LWR-PI^2, the meta basis function of LWR-PI^2 is evenly distributed in the time dimension. Actually, it is impossible to go through two points if the time indexes of these points are distributed in the same basis function. So, we can not easily get a trajectory which could via two certain points unless the distribution of the basis functions can be changed adaptively to meet the task's requirements. As shown in Fig.12, we find out that using LWR-PI^2 can not make the trajectory pass through the symbol 'x' in time. However guided by iLWR-PI^2-AL, the robot can pass through the given points at a specific time.

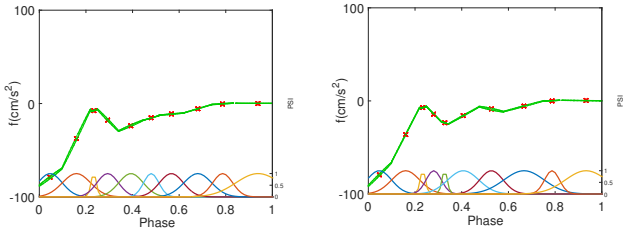
In Fig.13(a), the shape and scatter of meta basis function generated from imitation learning sounds arbitrary to some degree. Even more, after several iterations of ALTS, the meta kernels nearby 0.3067 to 0.4267 in phase is evolving into three distinct meta basis functions in Fig.13(b).

G. Application Scenario III: UR5

In this part, we employ UR5 robot shown to verify the effectiveness of the proposed method. There are six joints on UR5 , and from A to F it successively represents as: base, Shoulder, Elbow, wrist1, wrist2 and wrist3.

The robot is supposed to move within given duration $5s$, and we expect the arm can move via the landmark with various time index form $1.25s$ to $1.95s$ by self-learning and self-improved.

In the experiment, we set the start point with (83.88,-175.09,601.31) mm (unit will be omitted later)in the operation



(a) The distribution of meta basis functions by using iLWR-PI^2-AL under first recombination (b) The distribution of meta basis functions by using iLMR-PI^2-AL under final recombination

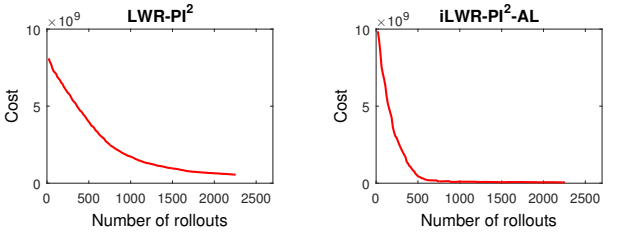
Fig. 13. Adaptive evolving of meta basis functions II

space, and the terminal point is (91.23,-630.98,-296.22) and the choosed landmark is (-325.92,-552.71,231.54).Our model is learned in joint space and we can get it by inverse kinematics equations.

We now proceeded to test the performance of iLWR-PI^2-AL, iLWR-PI^2 and LWR-PI^2. Detailed results are listed in Table.III. And the associated cost function is carefully designed in the form similar to equation (31) with D equals 6. Specifically, we execute the experiment fifteen times with the same via point and different time index. According to the result, the proposed iLWR-PI^2-AL methods performs better than original LWR-PI^2 and iLWR-PI^2 .

TABLE III
THE FINAL COSTS WHEN ALGORITHM STOPS

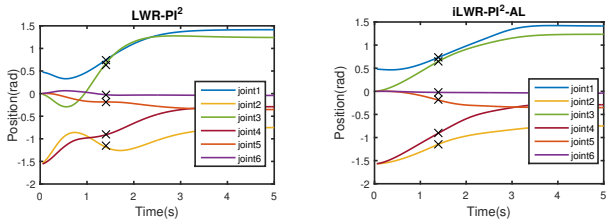
Time	Cost$_1$(LWR-PI^2)	Cost$_2$(iLWR-PI^2)	Cost$_3$(iLWR-PI^2-AL)
1.25	6.35E+08	1.70E+08	6.55E+07
1.30	5.63E+08	1.73E+08	6.43E+07
1.35	5.15E+08	1.74E+08	7.03E+07
1.40	4.60E+08	1.69E+08	7.19E+07
1.50	3.78E+08	1.75E+08	8.15E+07
1.55	3.57E+08	1.79E+08	6.79E+07
1.60	3.65E+08	1.73E+08	7.10E+07
1.65	3.61E+08	1.92E+08	8.66E+07
1.70	3.71E+08	1.87E+08	8.72E+07
1.75	3.62E+08	1.79E+08	8.14E+07
1.80	3.56E+08	1.80E+08	9.11E+07
1.85	3.63E+08	1.83E+08	9.36E+07
1.90	3.60E+08	1.78E+08	9.47E+07
1.95	3.63E+08	2.00E+08	8.38E+07
Mean	4.11E+08	1.81E+08	8.11E+07



(a) The learning curve of LWR-PI^2. (b) The learning curve of iLWR-PI^2-AL.

Fig. 14. Comparison of learning speed of LWR-PI^2 and iLWR-PI^2-AL

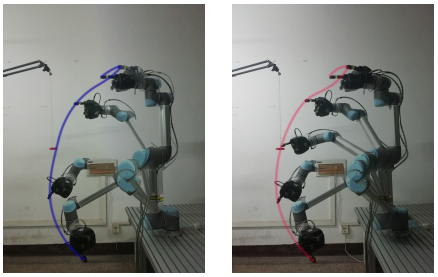
Besides, we compare the typical learning rate between iLWR-PI^2-AL and LWR-PI^2 shown in Fig.14. Also Fig.15



(a) the curve of LWR-PI^2 through via-point (b) the curve of iLWR-PI^2-AL through via-point

Fig. 15. the curve of iLWR-PI^2 and LWR-PI^2 through via-point in the joint space

shows the results of iLWR-PI^2-AL and LWR-PI^2 through via-point in the joint space. They both have excellent results when they pass the via-point. But clearly, the curve of iLWR-PI^2-AL shown in the Fig.15(b) is more smooth than LWR-PI^2, and the iLWR-PI^2-AL is more accurate than the LWR-PI^2 which can be clearly seen in joint2 of UR5. The actual track of iLWR-PI^2-AL and LWR-PI^2 on the UR5 arm is displayed in Fig.16. Compared Fig.16(a) with Fig.16(b), it can be noticed that the track of iLWR-PI^2-AL can hit the red landmark with pinpoint accuracy, but the track of LWR-PI^2 can only brush the landmark by inches. So we can conclude that our proposed iLWR-PI^2-AL has stronger self-learning ability.



(a) the track of LWR-PI^2 (b) the track of iLWR-PI^2-AL

Fig. 16. the motion track of the UR5

VI. Conclusion And Future Work

Generating new motor skills for robots attracts broad recent research interests and has wide applications in various applications. Nevertheless, the associated difficulty in skill transfer, how to endow the robot both flexible and specific imitation capability simultaneously, is still remaining. This paper proposes a new iLWR-PI^2-AL motor skill learning method to deal with this difficulty in all three phases of LfDRL. iLWR-PI^2-AL comprises two parts: DMPs-iLWR for LfD, and PI^2-DP for RL. Furthermore, ALTS is capable of capture the key features and conduct the target-oriented exploration based on the basis function created via previous ALTS. With this iterative process going for a certain time, the robot is then able to generate new skills to accomplish the new/unfamiliar tasks with an optimal/suboptimal criterion.

Future work can be focused on the following interesting aspects. It is worthwhile to investigate a more efficient way

to find the optimal hyper-parameters for PI^2 with less computational cost. Another potential improvement of the proposed method maybe achieved through not only considering the last information from the algorithm but also integrating the old information to get better flexibility for the algorithm. Finally, how to biologically-inspired deal with and manipulate the different basis functions generated by the ALTS in different evolving phases during the skill transfer is also an interesting future topic [35].

APPENDIX

Algorithm 2 Basis_Learning

Require: trajectories $\xi_d^\star$, number of basis function Γ
Ensure: weight $\mathbf{w}_d$, basis function $\mathbf{B}_d$

$\{\boldsymbol{\mu}_j\}_1^\Gamma \leftarrow$ DB-KMEANS($\xi_d^\star, \Gamma$) // decide initial cluster centers
$\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}_1^\Gamma \leftarrow$EM-GMM($\xi_d^\star, \Gamma$) // search for cluster centers and variances
$\{c_j, \sigma_j\}_1^\Gamma \leftarrow$PROJ($\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}_1^\Gamma$) // projected onto phase-axis to obtain parameters of meta basis function
for $j = 1$ to Γ **do**
  $\gamma^{(j)} \leftarrow$ Equ.5($\{c_j, \sigma_j\}_1^\Gamma$) // equation 5
  $\mathbf{B}_d^{(j)} \leftarrow s_t \gamma^{(j)}$
  $\mathbf{B}_d^{(j+\Gamma)} \leftarrow \gamma^{(j)}$
end for
$\mathbf{B}_d \leftarrow \{\mathbf{B}_d^j\}_1^{2*\Gamma}$ // assembled into basis in term of DOF
$\mathbf{w}_d \leftarrow$iLWR_LfD($\xi_d^\star, \mathbf{B}_d$) // conduct imitation learning
return $\mathbf{w}_d, \mathbf{B}_d$

Algorithm 3 Weight_Learning

Require: weight $\mathbf{w}^{\text{old}}$, basis function $\mathbf{B}$
Ensure: new weight $\mathbf{w}^{\text{new}}$, two-tuple set for sample trajectories and respective rewards$\{\xi^k, R(\xi^k)\}_1^{\text{K}}$

for $k = 1$ to K **do**
  for $d = 1$ to D **do**
    for $i = 1$ to N **do**
      $S(\xi_{d;i;k}) \leftarrow$ Equ.15($\mathbf{w}^{\text{old}}, \mathbf{B}$) //equation 15
      $P(\xi_{\cdot;i;k}) \leftarrow$ Equ.19($S(\xi_{d;i;k})$) //equation 19
      $u(\xi_{d;i;k}) \leftarrow$ Equ.18($S(\xi_{d;i;k})$) //equation 18
    end for
  end for
end for
$\xi^k \leftarrow \{\xi_{d;\text{N};k}\}_1^{\text{D}}$
$R\left(\xi^k\right) = \sum_{d=1}^{D} S\left(\xi_{d;\text{N};k}\right)$
for $d = 1$ to D **do**
  for $i = 1$ to N **do**
    $\mathbf{w}_{d;i} \leftarrow$ Equ.16($P(\xi_{\cdot;i;k}), u(\xi_{d;i;k})$)) //equation 16
    $\Delta\mathbf{w}_{d;i} = \mathbf{w}_{d;i} - \mathbf{w}_d$
  end for
  $\Delta\mathbf{w}_d \leftarrow$ Equ.20($\mathbf{B}, \Delta\mathbf{w}_{d;i}$) //equation 20
  $\mathbf{w}_d = \Delta\mathbf{w}_d + \mathbf{w}_d^{\text{old}}$
end for
$\mathbf{w}^{\text{new}} \leftarrow \{\mathbf{w}_d\}_1^{\text{D}}$
return $\mathbf{w}^{\text{new}}, \{\xi^k, R(\xi^k)\}_1^{\text{K}}$

Algorithm 4 Policy_Evaluation

Require: weight $\mathbf{w}$, basis function $\mathbf{B}$
Ensure: trajectory ξ, reward $R(\xi)$

$\{\mathbf{w}_d\}_1^{\text{D}} \leftarrow \mathbf{w}$ //resolved into each DOF
$\{\mathbf{B}_d\}_1^{\text{D}} \leftarrow \mathbf{B}$ //resolved into each DOF
for $d = 1$ to D **do**
  $\xi_d \leftarrow$ Equ.6($\mathbf{w}_d, \mathbf{B}_d$) //equation 6
end for
$R(\xi) \leftarrow$Equ.7$\left(\{\xi_d\}_1^{\text{D}}\right)$ //equation 7
$\xi \leftarrow \{\xi_d\}_1^{\text{D}}$
return $\xi, R(\xi)$

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[2] J. Rey, K. Kronander, F. Farshidian, J. Buchli, and A. Billard, "Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies," *Autonomous Robots*, vol. 42, no. 1, pp. 45–64, 2018.

[3] M. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Journal of Intelligent and Robotic Systems*, vol. 15, no. 1, pp. 1–2, 2013.

[4] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgtter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 145–157, 2012.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[6] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 29–31 Oct 2018, pp. 651–673.

[7] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2042–2062, 2018.

[8] U. E. Ogenyi, J. Liu, C. Yang, Z. Ju, and H. Liu, "Physical human-robot collaboration: Robotic systems, learning methods, collaborative strategies, sensors, and actuators," *IEEE transactions on cybernetics*, 2019.

[9] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[10] A. Ude and T. Gams, A. Asfour, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[11] C. Yang, C. Zeng, Y. Cong, N. Wang, and M. Wang, "A learning framework of adaptive manipulative skills from human to robot," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1153–1161, 2019.

[12] S. M. Khansari-Zadeh and A. Billard, "Bm: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models," in *2010 IEEE Int. Conf. Robotics and Automation*, Anchorage,USA, May 2010, pp. 2381–2388.

[13] S. Manschitz, M. Gienger, J. Kober, and J. Peters, "Mixture of attractors: A novel movement primitive representation for learning motor skills from demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 926–933, 2018.

[14] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 777–787, 2019.

[15] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1, pp. 11–73, 1997.

[16] O. Sigaud, C. Salaun, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: A survey," *Robotics and Autonomous Syst.*, vol. 59, no. 12, pp. 1115–1129, 2011.

[17] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, 2015.

[18] H. Ben Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014/5/31 - 2014/6/7, pp. 2831–2837.

[19] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011.

[20] J. Macedo, C. Santos, and L. Costa, "Using cost-regularized kernel regression with a high number of samples," in *2014 IEEE Int. Conf. Autonomous Robot Syst. and Competitions*, Espinho, Portugal, May 2014, Book Section, pp. 1371–1394.

[21] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *22th Int. Joint Conf. Artificial Intell.*, Barcelona,Spain, July 2011, pp. 2650–2655.

[22] M. D. Gregory, S. V. Martin, and D. H. Werner, "Improved electromagnetics optimization: The covariance matrix adaptation evolutionary strategy." *IEEE Antennas Propagat. Mag.*, vol. 57, no. 3, pp. 48–59, Jun. 2015.

[23] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. P. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, 2016.

[25] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, 2010.

[26] J. Fu, L. Ning, S. Wei, and L. Zhang, "A novel ds-gmr coupled primitive for robotic motion skill learning," in *2015 Int. Conf. Ind. Informatics-Computing Technology, Intelligent Technology, Ind. Inform. Integration*, Wuhan, China, Dec 2015, pp. 111–115.

[27] J. Fu, S. Chen, M. Pang, and P. Lou, "Robot motor skill acquisition with alternate learning in two spaces," *Journal of Huazhong University of Science and Technology(Natural Science Edition)*, vol. 45, no. 10, pp. 90–94+110, 2017.

[28] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal.Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[30] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control (Advanced Textbooks in Control and Signal Processing)*, 1st ed. Springer, 2 2011.

[31] R. Pfeifer and J. Bongard, *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

[32] A. Karni, G. Meyer, P. Jezzard, M. M. Adams, R. Turner, and L. G. Ungerleider, "Functional mri evidence for adult motor cortex plasticity during motor skill learning," *Nature*, vol. 377, no. 6545, pp. 155–158, 1995.

[33] U. Halsband and R. K. Lange", "Motor learning in man: A review of functional and clinical studies," *Journal of Physiology-Paris*, vol. 99, no. 4, pp. 414 – 424, 2006.

[34] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC, Mar. 1994.

[35] C. Yang, C. Chen, N. Wang, Z. Ju, J. Fu, and M. Wang, "Biologically inspired motion modeling and neural control for robot learning from demonstrations," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 2, pp. 281–291, 2018.