applied sciences

MDPI

Article

Compound Heuristic Information Guided Policy Improvement for Robot Motor Skill Acquisition

Jian Fu [1,*], Cong Li [1], Xiang Teng [1], Fan Luo [1] and Boqun Li [2]

[1] School of Automation, Wuhan University of Technology, Wuhan 430070, China; lc_own@whut.edu.cn (C.L.); tengxiang@whut.edu.cn (X.T.); dr_luofan@whut.edu.cn (F.L.)

[2] School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114051, China; lbqhylyx@ustl.edu.cn

* Correspondence: fujian@whut.edu.cn

Received: 30 June 2020; Accepted: 30 July 2020; Published: 3 August 2020

Abstract: Discovering the implicit pattern and using it as heuristic information to guide the policy search is one of the core factors to speed up the procedure of robot motor skill acquisition. This paper proposes a compound heuristic information guided reinforcement learning algorithm PI^2-CMA-KCCA for policy improvement. Its structure and workflow are similar to a double closed-loop control system. The outer loop realized by Kernel Canonical Correlation Analysis (KCCA) infers the implicit nonlinear heuristic information between the joints of the robot. In addition, the inner loop operated by Covariance Matrix Adaptation (CMA) discovers the hidden linear correlations between the basis functions within the joint of the robot. These patterns which are good for learning the new task can automatically determine the mean and variance of the exploring perturbation for Path Integral Policy Improvement (PI^2). Compared with classical PI^2, PI^2-CMA, and PI^2-KCCA, PI^2-CMA-KCCA can not only endow the robot with the ability to realize transfer learning of trajectory planning from the demonstration to the new task, but also complete it more efficiently. The classical via-point experiments based on SCARA and Swayer robots have validated that the proposed method has fast learning convergence and can find a solution for the new task.

Keywords: robot learning; reinforcement learning; heuristic information; KCCA; PI^2-CMA

1. Introduction

Imitation learning (IL) and reinforcement learning (RL) [1] have always been a hot topic in the field of robot skill acquisition. Imitation learning can be divided into two categories: behavioral cloning (BC) and inverse reinforcement learning (IRL). BC is the method of learning expected policy directly from expert teaching information, while IRL learns policy indirectly using reward function. Hidden Markov Model (HMM) [2], Dynamic Movement Primitives (DMPs), Probabilistic Movement Primitives (ProMPs) [3], Dynamic Systems (DS) [4], and Cross Entropy Regression (CER) are popular Behavior Cloning methods. The most frequently used methods of IRL are Maximum Margin Planning (MMP) [5] and Markov Process (MP).

Imitation learning is limited because it requires the robot to learn only from demonstrated trajectories. When the reproduction environment is different from the demonstration environment or there is a big deviation, such as placing an obstacle on the path of the robot, the imitation learning method may fail. Instead, RL allows the robot to find a new control policy by exploring the state-action space freely. The combination of IL and RL aims to use the advantages of two methods to overcome their respective shortcomings, so that the robot can adapt to the deviation from the demonstration behavior, so as to improve the performance of the robot.

The classical RL methods include SARSA [6], Natural Actor-Critic (NAC) [7], Policy Learning by Weighting Exploration with the Returns (PoWER) [8], Relative Entropy Policy Search (REPS) [9], and Path Integral Policy Improvement (PI^2). We believe that PI^2 [10] is one of the most effective, numerically robust, and easy reinforcement algorithms. However, classical PI^2 searches the whole parameter space, so it is less efficient to complete the task. Kernel Canonical Correlation Analysis (KCCA) can infer the implicit nonlinear heuristic information between the joints of the robot when facing the new task, leading PI^2 to find a solution. Based on the works of Covariance Matrix Adaptation (CMA) [11] together with our previous research on KCCA [12], we propose a new algorithm PI^2-CMA-KCCA in this paper, where KCCA and CMA are integrated as compound heuristic information to speed up the learning procedure from the demonstration to a new task.

This paper is structured into several major sections: Section 2 investigates the DMP used in the imitation learning phase. In this paper, we use Dynamic Movement Primitives as the underlying policy representation. Section 3 briefly introduces the algorithm PI^2-CMA which aims at discovering the hidden relationship between the components of the weight. In Section 4, we introduce the algorithm KCCA, and finally derive the algorithm PI^2-CMA-KCCA. In Section 5, based on SCARA and Swayer robots, we validate our algorithm through the classical via-point task and analyze the experimental results. Finally, the conclusions are given in Section 6.

2. Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) have been used in many disciplines to model complex behaviors. In this paper, we use DMPs as the underlying policy representation. A joint of the robot can be regarded as a DMP. The DMP [13] consists of a damped spring system and a learnable nonlinear forcing term, by which the desired behavior of the joint can be obtained. Its expression is given by:

$$\begin{cases} \zeta^2 \ddot{y}_t = \underbrace{\alpha_y(\beta_y(g - y_t) - \zeta \dot{y}_t)}_{\alpha_z} + \underbrace{h(x_t)x_t(g - y_0)}_{\alpha_f} \\ \zeta \dot{x}_t = -\alpha_x x_t \\ h(x_t) = \frac{\sum_{i=1}^{M} \Psi_i(x)\omega_i}{\sum_{i=1}^{M} \Psi_i(x)} = \sum_{i=1}^{M} \Psi_i \omega_i = \boldsymbol{\Psi}\boldsymbol{\omega}, \end{cases} \tag{1}$$

where, if the forced term $\alpha_f = 0$, α_z represents a globally stable second-order damped spring system. ζ is a time constant and represents the proportional coefficient of the duration of the motion. g is the target position. y_0 is the initial position, and variable y_t would be interpreted as the desired position of the joint. $h(x_t)$ is a fitting function, and x_t can be conceived of as a phase variable. $\Psi_i(x)$ represents the ith basis function $\Psi_i(x) = \exp(-(\mathrm{x_t} - \mathrm{c_i})^2/2\sigma_i^2)$, where c_i and σ_i are constants that determine the width and center of the basis function respectively. w_i is the weight corresponding to the ith basis function. M is the number of exponential basis functions. $\boldsymbol{\omega} \in \mathbb{R}^{M\times 1}$ is the weight vector. α_y, β_y can be obtained by adjusting the damped spring system to the second-order critical damping system.

The parameter ω_i of DMP is learned by Locally Weighted Regression algorithm (LWR). The algorithm will find the corresponding ω_i for each basis function Ψ_i by minimizing the cost function S. The function is defined by:

$$S(\boldsymbol{\omega}) = \sum_{j=1}^{N} \left(\sum_{i=1}^{M} \Psi_i \left(x^j \right) \omega_i - h \left(x^j \right) \right)^2, \tag{2}$$

where $(x^j, h\left(x^j\right))$ is the jth sample. Obviously, ω_i cooperates with each other in linear combination patterns because of the introduction of basis functions. Thus, we can optimize them by means of linear correlation techniques. A summary of the notation frequently used in this article is given by Table 1.

Table 1. Summary of the notation frequently used in the article.

Symbol	Definition
$\ddot{y}_t, \dot{y}_t, y_t$	The desired acceleration, velocity, position of the joint
$h(x_t)$	Fitting function. x_t is a phase variable
$P_{.,k,i}$	The probability-weighted value of the kth trajectory of the dth DOF at time i
$\delta\boldsymbol{\omega}_{d,i}$	The dth joint's correction weight vector at time i
$\delta\boldsymbol{\omega}_d$	Average value of $\delta\boldsymbol{\omega}_{d,i}$ over time
$\boldsymbol{\omega}_d$	The dth joint's weight vector
$\check{\boldsymbol{\omega}}_d$	The dth joint's weight vector with perturbation
$\tilde{\boldsymbol{\omega}}$	$[\check{\boldsymbol{\omega}}_1^T, \check{\boldsymbol{\omega}}_2^T, \cdots, \check{\boldsymbol{\omega}}_D^T]^T$
$\phi(\cdot)$	Mapping function. Mapping a variable to V-dimensional space
$\check{\boldsymbol{\omega}}_d^s$	The weight perturbation sample of the dth joint, i.e., $\check{\boldsymbol{\omega}}_d^s = \{\check{\boldsymbol{\omega}}_{d,1}, \check{\boldsymbol{\omega}}_{d,2}, \cdots, \check{\boldsymbol{\omega}}_{d,N}\}$
J_d	The dth joint's cost for given task
J^p	The total cost of D joints at the pth iteration
$K(\cdot)$	Kernel matrix
T_r	Decline rate of cost between J^{p-1} and J^p
$\{J_d, \delta\boldsymbol{\omega}_{d,i}\}_K$	J_d and $\delta\boldsymbol{\omega}_{d,i}$ which are calculated from K roll-outs
$\{J_d, \delta\boldsymbol{\omega}_{d,i}\}_{Ke}$	J_d and $\delta\boldsymbol{\omega}_{d,i}$ which are calculated from Ke elite roll-outs, and Ke elite roll-outs are obtained by sorting K roll-outs according to the cost

3. Path Integral Policy Improvement with Covariance Matrix Adaption

PI^2 is derived from the first principles of stochastic optimal control. What sets PI^2 apart from other direct policy improvement algorithms is its use of probability-weighted averaging to perform a parameter update, rather than using an estimate of the gradient. In Section 2, we obtain the weight vector $\boldsymbol{\omega}$ by parameterizing the demonstration trajectory. The idea of PI^2 is to add stochastic exploration noise $\boldsymbol{\varepsilon}_d$ ($\boldsymbol{\varepsilon}_d \sim \mathcal{N}(\mathbf{0}, \sigma^2)$) to $\boldsymbol{\omega}_d$ and generate K roll-outs with different costs by executing parameterized policy. As for a robot with D DOF, the cost function of the kth roll-out at time i is

$$J(\tau_{.,i,k}) = \sum_{d=1}^{D}\left[\varphi_{d,N,k} + \sum_{j=i}^{N-1} q_{d,j,k} + \frac{1}{2}\sum_{j=i}^{N-1}(\boldsymbol{\omega}_d + \boldsymbol{M}_{d,j,k}\boldsymbol{\varepsilon}_{d,j,k})^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{\omega}_d + \boldsymbol{M}_{d,j,k}\boldsymbol{\varepsilon}_{d,j,k})\right], \tag{3}$$

where $\tau_{.,i,k}\tau$ is a sample path (or trajectory piece). $\cdot$ indicates all DOF. $\varphi_{d,N,k}$ represents the terminal reward of the kth trajectory of the dth DOF. $q_{d,j,k}$ is the immediate reward of the kth trajectory of the dth DOF at time j. Specifically, it is expressed as $\ddot{y}_t$ (i.e., the square of joint's acceleration). $\boldsymbol{M}_{d,j,k}$ is the mapping matrix of the kth trajectory of the dth DOF at time j. $\boldsymbol{R}$ is the positive semi-definite weight matrix of the quadratic control cost. N is the maximum value of the time index.

Next, the exploration is evaluated. First, the cost of obtained trajectories is sorted, then K_e elite samples are selected, and finally perform probability-weighted averaging to obtain $\delta\boldsymbol{\omega}_{d,i}$ with th DOF at time i:

$$\delta\boldsymbol{\omega}_{d,i} = \sum_{k=1}^{K_e}\left[P(\tau_{.,i,k})\boldsymbol{M}_{d,i,k}\boldsymbol{\varepsilon}_{d,i,k}\right], \tag{4}$$

where $P(\tau_{.,i,k})$ is the probability-weighted value of the kth trajectory of the dth DOF at time i. It is obtained by softmax transformation of the cost function:

$$P_{.,k,i} = P\left(\tau_{.,i,k}\right) = \frac{e^{-\frac{1}{\lambda}\left(J\left(\tau_{.,i,k}\right)\right)}}{\sum_{k=1}^{K_e} e^{-\frac{1}{\lambda}\left(J\left(\tau_{.,i,k}\right)\right)}}, \tag{5}$$

where λ is an appropriate constant.

Later, $\delta\boldsymbol{\omega}_{d,i}$ is averaged over time to get $\delta\boldsymbol{\omega}_d$ further to obtain the new weight (i.e., $\boldsymbol{\omega}_d^{\text{new}} = \boldsymbol{\omega}_d^{\text{old}} + \delta\boldsymbol{\omega}_d$). By searching the parameter space iteratively, PI^2 will eventually find a solution for the new task. Classical PI^2 only updates the mean $\boldsymbol{\omega}$, and the covariance $\boldsymbol{\sigma}^2$ is a constant ($\boldsymbol{\sigma}^2 = \lambda_{init}\mathbf{I}_M$). M is the number of base functions. λ_{init} determines the magnitude of initial exploration noise. PI^2-CMA aims to determine the magnitude of the exploration noise automatically and to infer the implicit linear correlation between the basis functions within the joint of the robot. In other words, covariance matrix adaption is expressed as:

$$\Sigma_{d,i}^{new} = \sum_{k=1}^{K_e} P_{.,k,i}(\delta\boldsymbol{\omega}_{d,i})(\delta\boldsymbol{\omega}_{d,i})^{\mathrm{T}}. \tag{6}$$

In addition, the covariance update equation is

$$\Sigma_d^{new} = \frac{\sum_{i=1}^{N}(N-i)\Sigma_{d,i}^{new}}{\sum_{i=l}^{N}(N-l)}. \tag{7}$$

As we can see that the vanilla PI^2-CMA only infers linear correlation of weight within a DMP independently.

4. PI^2-CMA with Kernel Canonical Correlation Analysis

In Equation (3), it can be seen that the perturbation $\boldsymbol{\varepsilon}$ of PI^2 is generated with equal probability for each joint's weight vector $\boldsymbol{\omega}_d$. The PI^2-CMA takes into account the implicit linear correlations between the basis functions of the joint and automatically updates the covariance. However, when a task is assigned to a multi-joint robot , there exists an unknown hidden task-oriented pattern between weight vector with perturbation $\check{\boldsymbol{\omega}}_{d_i}$ and $\check{\boldsymbol{\omega}}_{d_j}$ ($\check{\boldsymbol{\omega}}_{d_k} = \boldsymbol{\omega}_{d_k} + \delta\boldsymbol{\omega}_{d_k}, k \in \{i, j\}$). The pattern can be inferred from experience and expressed as a nonlinear correlation. Specifically, we can apply Kernel Canonical Correlation Analysis (KCCA) to infer the implicit nonlinear heuristic information between the joints of the robot to guide PI^2 to a search policy for the new task of trajectory planning. In the paper, we not only consider the linear correlations between the perturbation vector's components within a DMP but also take nonlinear correlations of the perturbation vector between DMPs into account to expedite the learning procedure of the robot.

4.1. Nonlinear Correlation Heuristic Information

According to Equations (3) and (4), perturbation is generated with equal probability for each joint's weight vector. In other words, there is no correlation between the joints. Therefore, $[\check{\boldsymbol{\omega}}_1^T, \check{\boldsymbol{\omega}}_2^T, \cdots, \check{\boldsymbol{\omega}}_D^T]^T \in \mathbb{R}^{DM\times 1}$ is denoted as $\tilde{\boldsymbol{\omega}}$, and covariance matrix of $\tilde{\boldsymbol{\omega}}$ is given by:

$$\Sigma_{\tilde{\omega}} = diag[\boldsymbol{\sigma}_1^2, \boldsymbol{\sigma}_2^2, \cdots, \boldsymbol{\sigma}_D^2] \in \mathbb{R}^{DM\times DM}, \tag{8}$$

where $diag[\cdot]$ represents diagonal matrix and $\boldsymbol{\sigma}_1^2 = \cdots = \boldsymbol{\sigma}_D^2 = \boldsymbol{\sigma}^2$.

When the multi-joint robot completes the task, we believe that there are some hidden patterns between the joints. The implicit patterns between the perturbation vectors of the multi-joint robot are expressed as nonlinear correlations. With the help of the kernel method, the dth joint's perturbation vector $\check{\boldsymbol{\omega}}_d \in \mathbb{R}^{M\times 1}$ is mapped to high-dimensional feature space $\phi(\check{\boldsymbol{\omega}}_d) \in \mathbb{R}^{V\times 1}$. $\tilde{\boldsymbol{\omega}}$ is also mapped to high-dimensional feature space $\phi(\tilde{\boldsymbol{\omega}}) \in \mathbb{R}^{DV\times 1}$, and then covariance matrix of $\tilde{\boldsymbol{\omega}}$ can be described as:

$$\Sigma_{\phi(\tilde{\omega})}=\begin{bmatrix} \Gamma(\breve{\omega}_1,\breve{\omega}_1) & \Gamma(\breve{\omega}_1,\breve{\omega}_2) & \cdots & \Gamma(\breve{\omega}_1,\breve{\omega}_D) \\ \Gamma(\breve{\omega}_2,\breve{\omega}_1) & \Gamma(\breve{\omega}_2,\breve{\omega}_2) & \cdots & \Gamma(\breve{\omega}_2,\breve{\omega}_D) \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma(\breve{\omega}_D,\breve{\omega}_1) & \Gamma(\breve{\omega}_D,\breve{\omega}_2) & \cdots & \Gamma(\breve{\omega}_D,\breve{\omega}_D) \end{bmatrix}, \tag{9}$$

where $\Gamma(\breve{\omega}_{d_i},\breve{\omega}_{d_j}) = cov(\phi(\breve{\omega}_{d_i}),\phi(\breve{\omega}_{d_j}))$ is the covariance of $\breve{\omega}_{d_i}$ and $\breve{\omega}_{d_j}$ projected on high-dimensional space, expressing the implicit nonlinear pattern between the perturbation vectors of the d_ith joint and the d_jth joint . The intuitive policy is to obtain $\Sigma_{\phi(\tilde{\omega})}$ based on empirical samples, and then use $\Gamma(\breve{\omega}_{d_i},\breve{\omega}_{d_j})$ as the heuristic information. That is, given the perturbation vector $\breve{\omega}_{d_i}$, the exploration of $\breve{\omega}_{d_j}$ is guided by the covariance $\Gamma(\breve{\omega}_{d_i},\breve{\omega}_{d_j})$.

Note that $\Gamma(\breve{\omega}_{d_i},\breve{\omega}_{d_j})$ is in the unified space coordinate system (i.e., $\phi(\breve{\omega})$), but analyzing the correlation between the perturbation vector $\breve{\omega}_{d_i}$ of the d_ith joint and the perturbation vector $\breve{\omega}_{d_j}$ of the d_jth joint in a uniform coordinate system is not the best choice. Because after the perturbation vectors of different joints are mapped to a high-dimensional space, the correlation coefficients between the joints can be maximized only after a proper projection transformation, and such projection matrices are usually not necessarily identical. Here, the generalized Rayleigh Entropy is used to find the maximum correlation coefficient of $\phi(\breve{\omega}_{d_i})$ and $\phi(\breve{\omega}_{d_j})$, and Maximum Likelihood Estimation is used to infer the nonlinear correlation between $\breve{\omega}_{d_i}$ and $\breve{\omega}_{d_j}$. If $d_i \neq d_j$, $\Theta(\breve{\omega}_{d_i},\breve{\omega}_{d_j}) = cov(P_r\{\Phi(\breve{\omega}_{d_i})\}, P_r\{\Phi(\breve{\omega}_{d_j})\})$ can be the heuristic information, and P_r is the projection operator. That is, given the perturbation vector $\breve{\omega}_{d_i}$, $\breve{\omega}_{d_j}$ can be obtained from the covariance $\Theta(\breve{\omega}_{d_i},\breve{\omega}_{d_j})$. Equation (9) can now be expressed as follows:

$$\Sigma^{+}_{\phi(\tilde{\omega})}=\begin{bmatrix} \Gamma(\breve{\omega}_1,\breve{\omega}_1) & \Theta(\breve{\omega}_1,\breve{\omega}_2) & \cdots & \Theta(\breve{\omega}_1,\breve{\omega}_D) \\ \Theta(\breve{\omega}_2,\breve{\omega}_1) & \Gamma(\breve{\omega}_2,\breve{\omega}_2) & \cdots & \Theta(\breve{\omega}_2,\breve{\omega}_D) \\ \vdots & \vdots & \ddots & \vdots \\ \Theta(\breve{\omega}_D,\breve{\omega}_1) & \Theta(\breve{\omega}_D,\breve{\omega}_2) & \cdots & \Gamma(\breve{\omega}_D,\breve{\omega}_D) \end{bmatrix}. \tag{10}$$

4.2. Robot Intelligent Trajectory Inference with KCCA

KCCA [14] is a nonlinear correlation analysis method. In this paper, we employ KCCA on the elite samples from the robot's first joint to other joints, and make heuristic inference. After the pth iteration of PI^2-CMA, the program records the total reward $J^p(\tau)$ (one episodic samples), based on the current $\breve{\omega}^p_{1,\cdots,D}$ (perturbation vectors of D joints). Then, $J^p(\tau)$ is compared with the total reward $J^{p-1}(\tau)$ based on $\breve{\omega}^{p-1}_{1,\cdots,D}$ at the (p-1)th iteration to obtain a decline rate T_r.

If T_r is greater than its upper threshold T_{max}, T_r is updated to T_{max}, and the program executes KCCA learning. At this time, the weight perturbation sample $\breve{\omega}^s_1 = \{\breve{\omega}_{1,1},\breve{\omega}_{1,2},\cdots,\breve{\omega}_{1,N}\}$ on the first joint and the weight perturbation sample $\breve{\omega}^s_2 = \{\breve{\omega}_{2,1},\breve{\omega}_{2,2},\cdots,\breve{\omega}_{2,N}\}$ on the second joint in the pth iteration are respectively mapped in high dimensions to obtain $[\phi(\breve{\omega}_{d,1}) \quad \phi(\breve{\omega}_{d,2}) \quad \cdots \quad \phi(\breve{\omega}_{d,N})]$, denoted as $\Phi(\breve{\omega}^s_d) \in \mathbb{R}^{V\times N}(d \in [1,2])$. Next, find two sets of vectors $\boldsymbol{w_1}$, $\boldsymbol{w_2} \in \mathbb{R}^{V\times 1}$, so that the correlation coefficient between the data u and v after the projection of $\phi(\breve{\omega}_1)$ and $\phi(\breve{\omega}_2)$ is maximized. We will have:

$$\begin{aligned} u &= \boldsymbol{w}_1^T\phi(\breve{\omega}_1), \\ v &= \boldsymbol{w}_2^T\phi(\breve{\omega}_2). \end{aligned} \tag{11}$$

The covariance of u and v is given by:

$$var(u,v) = \frac{1}{N-1}\boldsymbol{w}_1^T\Phi(\breve{\omega}^s_1)\Phi(\breve{\omega}^s_2)^T\boldsymbol{w}_2. \tag{12}$$

Obviously, the vectors $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are located in the space spanned by data $\Phi(\check{\boldsymbol{\omega}}_1^s)$ and $\Phi(\check{\boldsymbol{\omega}}_2^s)$:

$$\begin{aligned} \boldsymbol{w}_1 &= \Phi(\check{\boldsymbol{\omega}}_1^s)\boldsymbol{\alpha_1}, \\ \boldsymbol{w}_2 &= \Phi(\check{\boldsymbol{\omega}}_2^s)\boldsymbol{\alpha_2}, \end{aligned} \tag{13}$$

where $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathbb{R}^{N\times 1}$. From Equations (12) and (13), correlation coefficient ρ can be obtained:

$$\begin{aligned} \rho &= \frac{\boldsymbol{\alpha}_1^T \Phi^T(\check{\boldsymbol{\omega}}_1^s)\Phi(\check{\boldsymbol{\omega}}_1^s)\Phi^T(\check{\boldsymbol{\omega}}_2^s)\Phi(\check{\boldsymbol{\omega}}_2^s)\boldsymbol{\alpha}_2}{\sqrt{\boldsymbol{\alpha}_1^T(\Phi^T(\check{\boldsymbol{\omega}}_1^s)\Phi(\check{\boldsymbol{\omega}}_1^s))^2\boldsymbol{\alpha}_1}\sqrt{\boldsymbol{\alpha}_2^T(\Phi^T(\check{\boldsymbol{\omega}}_2^s)\Phi(\check{\boldsymbol{\omega}}_2^s))^2\boldsymbol{\alpha}_2}} \\ &= \frac{\boldsymbol{\alpha}_1^T K_{\omega_1} K_{\omega_2}\boldsymbol{\alpha}_2}{\sqrt{\boldsymbol{\alpha}_1^T K_{\omega_1} K_{\omega_1}\boldsymbol{\alpha}_1}\sqrt{\boldsymbol{\alpha}_2^T K_{\omega_2} K_{\omega_2}\boldsymbol{\alpha}_2}}. \end{aligned} \tag{14}$$

Kernel method is introduced in Equation (14), where $K(\cdot)$ is a kernel matrix. Without loss of generality, we fixed the denominator $\boldsymbol{\alpha}_1^T K_{\omega_1} K_{\omega_1}\boldsymbol{\alpha}_1 = 1$, $\boldsymbol{\alpha}_2^T K_{\omega_2} K_{\omega_2}\boldsymbol{\alpha}_2 = 1$ of Equation (14) to find a suitable $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ to maximize $\boldsymbol{\alpha}_1^T K_{\omega_1} K_{\omega_2}\boldsymbol{\alpha}_2$. Construct the Lagrange function:

$$\begin{aligned} L =& \boldsymbol{\alpha}_1^T K_{\omega_1} K_{\omega_2}\boldsymbol{\alpha}_2 - \frac{\lambda_1}{2}(\boldsymbol{\alpha}_1^T K_{\omega_1} K_{\omega_1}\boldsymbol{\alpha}_1 - 1) \\ &- \frac{\lambda_2}{2}(\boldsymbol{\alpha}_2^T K_{\omega_2} K_{\omega_2}\boldsymbol{\alpha}_2 - 1). \end{aligned} \tag{15}$$

We take partial derivatives of α_1 and α_2 in Equation (15), and obtain:

$$\begin{cases} K_{\omega_2}\boldsymbol{\alpha}_2 = \lambda K_{\omega_1}\boldsymbol{\alpha}_1 \\ \lambda = 2\lambda_1 = 2\lambda_2 = \boldsymbol{\alpha}_1^T K_{\omega_1} K_{\omega_2}\boldsymbol{\alpha}_2. \end{cases} \tag{16}$$

According to Equation (16), the implicit correlation $\mathrm{P}_{ca}(\Theta(\check{\boldsymbol{\omega}}_1, \check{\boldsymbol{\omega}}_2))$ between the first joint and the second joint perturbation vector of the robot is obtained. When the current T_r is greater than the upper threshold T_{max}, $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2$ are recorded. Furthermore, Equation (15) can be repeatedly executed to obtain the implicit correlation between the first joint and the dth ($d \in [2, \cdots, D]$) joint. In this way, we get the implicit patterns $\{(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2), \cdots, (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_D)\}$ between the joints.

If T_r is less than the lower threshold T_{min}, the independent explorations of the whole joint are abandoned, and the KCCA prediction is turned on. That is, the first joint explores randomly to obtain a perturbation sample $\check{\boldsymbol{\omega}}_1^s$ and K_{ω_1} in the pth iteration, then $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$ is used to calculate K_{ω_2} of the second joint. Obviously, $\check{\boldsymbol{\omega}}_2$ can be computed from K_{ω_2}, and that will guide the exploration of the second joint. Repeat the same steps for the subsequent joints until the Dth joint.

4.3. The Combination of KCCA and CMA

We draw the schematic of the PI^2-CMA-KCCA's control flow as Figure 1, which works like a double closed-loop control system. Without loss of generality, there are D joints, and each joint corresponds to M basis functions, then $D \times M$ parameters need to be adjusted.

The outer loop aims at discovering the hidden patterns between the joints which are helpful to fulfill the new task. Specifically, we use the KCCA to infer the nonlinear correlations between the first joint and other joints for good declining rate, recorded as $\{(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2), \cdots, (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_D)\}$. After that, we can compute $\check{\boldsymbol{\omega}}_d$ from K_{ω_d} based on perturbation sample $\check{\boldsymbol{\omega}}_1^s$ and K_{ω_1}, and it will guide the perturbation of the dth joint as a means of exploring noise.

The inner loop extracts heuristic information from a corresponding single joint to guide the search. Specifically, in the previous iteration, it evaluates the cost J_d of K roll-outs (sorting and probability average), then selects the first K_e elites of the perturbation sample to obtain the covariance Σ_{ω_d} by the CMA algorithm. Furthermore, in current iteration , $\boldsymbol{\omega}_d \sim \mathcal{N}\left(\check{\boldsymbol{\omega}}_d, \Sigma_{\omega_d}\right)$ is used to explore the policy parameter $\boldsymbol{\omega}_d$. In other words, it applies compound heuristic information which integrates the inferred

hidden nonlinear patterns between the joints and the linear patterns within the joint to automatically determine the exploring proportion for each component of $\boldsymbol{\omega}_d$.

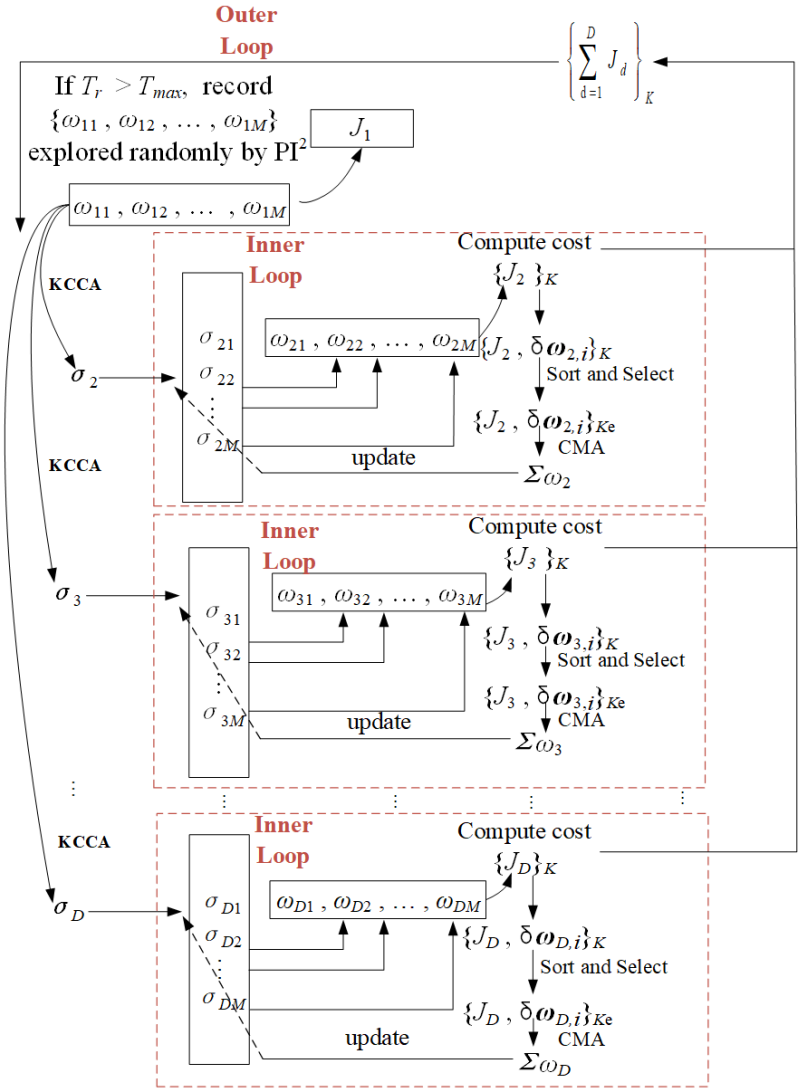

Figure 1. A typical system's architecture for PI^2-CMA-KCCA.

In short, the outer loop discovers and predicts the exploring mean of $\boldsymbol{\omega}_d$ for the dth ($d \neq 1$) DOF, the inner loop discloses and infers the exploring variance of $\boldsymbol{\omega}_d$ for the dth ($d \neq 1$) DOF. As for 1st DOF, a vanilla PI^2 is employed.

5. Evaluations

The cost function in our experiments is given by:

$$\begin{aligned} J =& 0.5 \times \sum_{d=1}^{D} \sum_{i=1}^{N-1} [10^7 (\ddot{y}^{(d)}(i))^2 + (a_f^{(d)}(i))^2] \\ &+ \sum_{d=1}^{D} 10^{11} (y^{(d)}(m) - y_v^{(d)})^2 \\ &+ \sum_{d=1}^{D} 10^3 [(\dot{y}^{(d)})^2 + (y^{(d)}(N) - y_g^{(d)})^2]. \end{aligned} \tag{17}$$

In Equation (17), D is the number of joints of SCARA and Swayer. $y^{(d)}(i), \dot{y}^{(d)}(i), \ddot{y}^{(d)}(i)$ denote the position, velocity, and acceleration of the dth joint of the robot at the time i. $y_v^{(d)}$ is the present point (i.e., via-point) to be passed through by the dth joint at time i. $y_g^{(d)}$ is the trajectory's end point of the d joint. $y^d(m)$ is the point passed through by the dth joint at time m. $a_f^{(d)}(i)$ is the acceleration of forcing term of the dth joint at time i. Equation (17) is a typical quadratic expression of total reward in the finite stage of reinforcement learning.

5.1. Passing through One Via-Point with SCARA

SCARA has three revolute joints q_1, q_2, q_3 and one prismatic joint q_4. In this experiment, the orientation of the end-effector of the robot arm is ignored, thus the SCARA robot arm can be regarded as a planar two-link mechanism.

The experiment is divided into the following four steps: **(1)** Set the Cartesian coordinate of the starting position of the end-effector to $(20, 0, 0)^{\mathrm{T}}$cm, and the corresponding joint angle is $(0, 0, 0)^{\mathrm{T}}$rad. Set the endpoint $(4.5, 16, 0)^{\mathrm{T}}$cm, and the corresponding joint angle is $(0.7068, 1.1796, 0)^{\mathrm{T}}$rad. **(2)** Drive the manipulator based on the principle of minimum jitter to obtain the demonstration. **(3)** Given a new task, use PI^2, PI^2-CMA, PI^2-KCCA, and PI^2-CMA-KCCA, respectively, to make the movement pass through a via-point at $m = 1.8$ s. **(4)** The four algorithms are iterated for 80 times respectively to gain new motor skills.

In this experiment, each of the four different algorithms is repeated 20 times. The best experimental results of PI^2, PI^2-CMA, PI^2-KCCA, and the five experimental results of PI^2-CMA-KCCA are selected for comparison. The optimization effect of PI^2-CMA-KCCA is analyzed by comparing the final cost of the system and the trajectories of joint space. It can be clearly seen from Figure 2 that PI^2-CMA-KCCA converges faster, and its final cost is the lowest (as shown in Table 2).

Table 2. The final cost of one via-point task with SCARA.

Algorithm	Final Cost
PI^2	2.2341×10^8
PI^2-CMA	1.7569×10^8
PI^2-KCCA	1.4089×10^8
PI^2-CMA-KCCA(test1)	1.0249×10^8
PI^2-CMA-KCCA(test2)	1.1095×10^8
PI^2-CMA-KCCA(test3)	1.0677×10^8
PI^2-CMA-KCCA(test4)	1.2106×10^8
PI^2-CMA-KCCA(test5)	1.0155×10^8

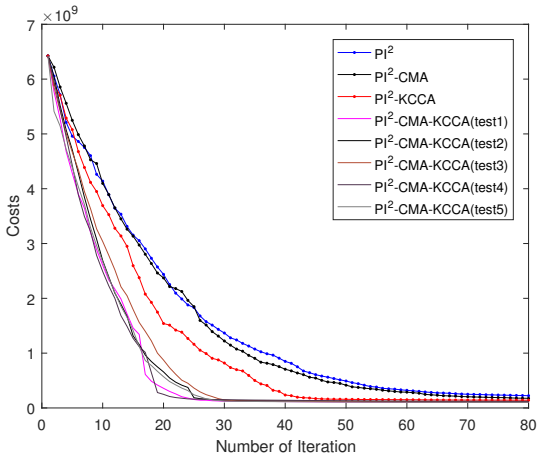

Figure 2. Cost of one via-point task.

Figure 3 shows the movement trajectories of the three joints of the SCARA in the joint space when the number of iterations is 35. Because the joint q_3 has no effect on the position of the end-effector, the value of q_3 in the joint space remains constant.

The blue line in Figure 3 is the trajectory of each joint of the robot arm under the PI^2-CMA-KCCA. Figure 3 demonstrates that only the blue lines pass through an intermediate via-point.

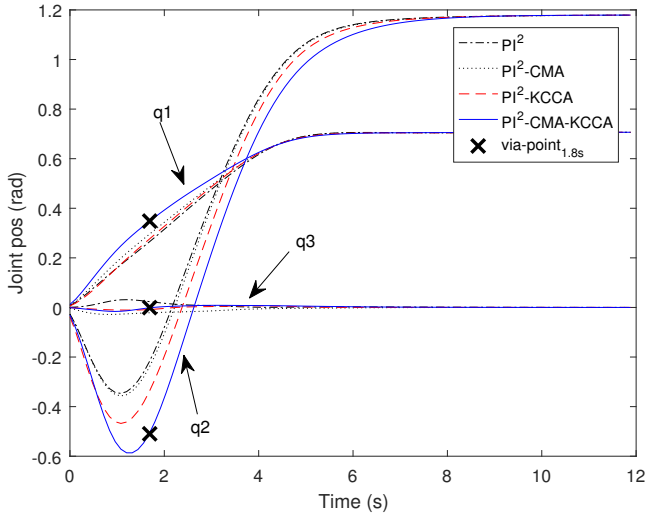

Figure 3. One via-point task when the number of iterations is 30.

5.2. Passing through Two Via-Point with SCARA

This experimental procedure is similar to Section 5.1, requiring the end-effector to pass the point $(18.2, 8.1, 0)^{\mathrm{T}}$ cm at $m = 1.8$ s and the point $(12.0, 13.5, 0)^{\mathrm{T}}$ cm at $m = 3.6$ s. Four different algorithms are also respectively repeated 20 times, and the best experimental results of PI^2, PI^2-CMA, PI^2-KCCA, and the five experimental results of PI^2-CMA-KCCA are selected for comparison. Figure 4 demonstrates that the PI^2-CMA-KCCA has better learning performance than the other algorithms. After 80 iterations, the final cost of the four algorithms is given in Table 3.

Figure 5 shows the trajectories of the three joints of the SCARA in the joint space when the number of iterations is 50, where the blue lines represent the joints' trajectories of the robot arm in joint space under the PI^2-CMA-KCCA. It demonstrates that, when the number of iterations is 50, the blue trajectories pass through two intermediate via-point at $t = 1.8$ s and $t = 3.6$ s accurately.

Table 3. The final cost of two via-point task with SCARA.

Algorithm	Final Cost
PI^2	7.1203×10^9
PI^2-CMA	5.5471×10^9
PI^2-KCCA	4.4999×10^9
PI^2-CMA-KCCA(test1)	2.7410×10^9
PI^2-CMA-KCCA(test2)	3.3489×10^9
PI^2-CMA-KCCA(test3)	3.4429×10^9
PI^2-CMA-KCCA(test4)	3.1620×10^9
PI^2-CMA-KCCA(test5)	3.0731×10^9

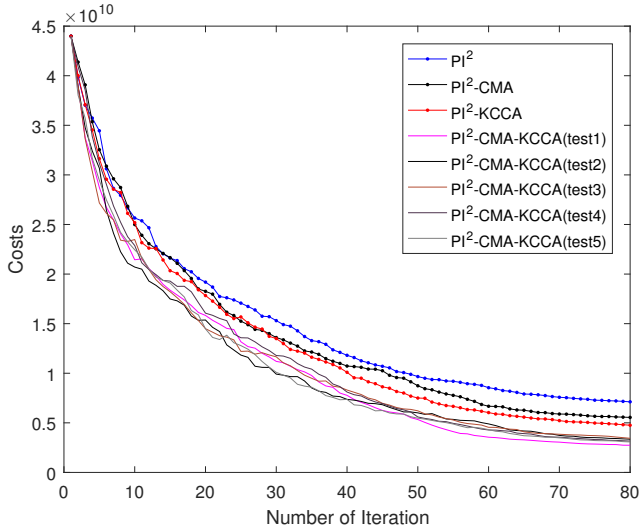

Figure 4. Cost of two via-point task.

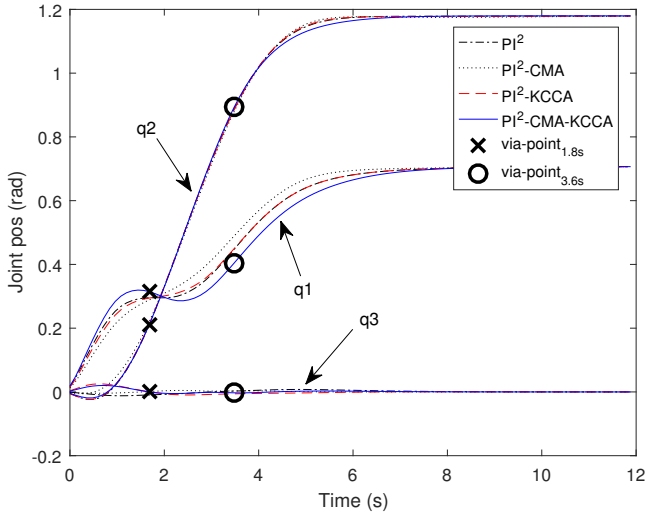

Figure 5. Two via-point task when the number of iterations is 50.

5.3. Passing Through One Via-Point with Swayer

Swayer is a lightweight collaborative robot created by Rethink Robotics. It has seven joints. In this subsection, q_i is used to represent the ith joint of Swayer. Because the q_7 has no effect on Cartesian position of the end-effector, only six joints are considered. This experiment uses the ROS platform to control the movement trajectory of the Swayer manipulator in the Ubuntu16 system, and validates the effectiveness of the PI^2-CMA-KCCA.

First, we set an arbitrary starting point $(697, 159, 514)^{\mathrm{T}}$ mm in the workspace of Swayer and an arbitrary endpoint $(300, -569, -65)^{\mathrm{T}}$ mm, then drag Swayer to record a demonstration from the starting point to the endpoint, then choose any reachable point $(840, -248, 595)^{\mathrm{T}}$ mm far away from the demonstration as a new task at $m = 1.2$ s. Finally, PI^2, PI^2-CMA, PI^2-KCCA, and PI^2-CMA-KCCA are repeated 20 times on Swayer. The five experimental results of the PI^2-CMA-KCCA and the best experimental results of the PI^2, PI^2-CMA and PI^2-KCCA are selected for analysis. Figure 6 shows the downward trend of the cost of the four algorithms, and the final cost of the four algorithms after 80 iterations is given in Table 4.

Table 4. The final cost of one via-point task with Swayer.

Algorithm	Final Cost
PI^2	7.1889×10^{10}
PI^2-CMA	5.0168×10^{10}
PI^2-KCCA	4.5494×10^{10}
PI^2-CMA-KCCA(test1)	2.2861×10^{10}
PI^2-CMA-KCCA(test2)	2.2581×10^{10}
PI^2-CMA-KCCA(test3)	2.2641×10^{10}
PI^2-CMA-KCCA(test4)	2.2554×10^{10}
PI^2-CMA-KCCA(test5)	2.2513×10^{10}

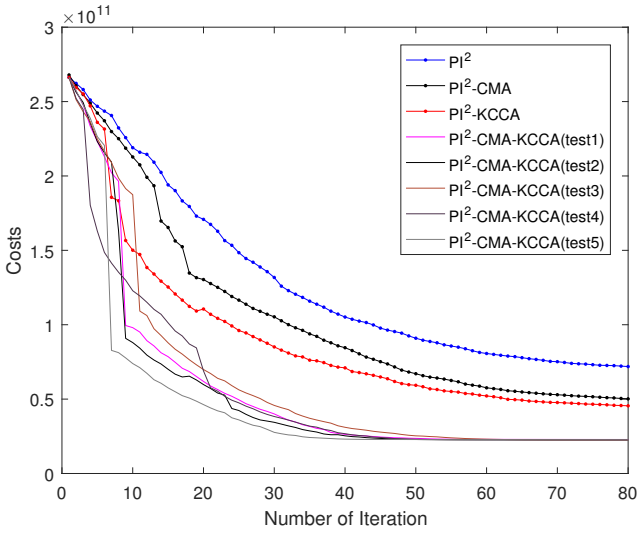

Figure 6. Cost of one via-point task.

Figure 6 illustrates that, during the reinforcement learning, learning efficiency of PI^2-CMA-KCCA is higher under the same number of iterations. At the same time, Figure 7 shows that the Swayer passes through an intermediate via-point accurately at $m = 1.2$ s when the number of iterations is 60.

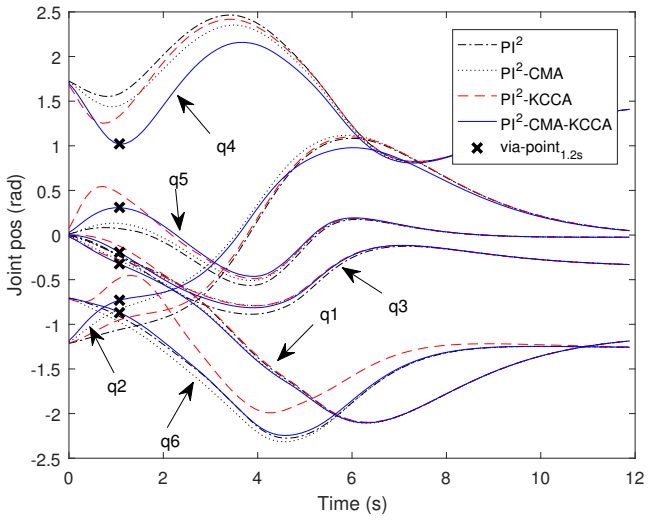

Figure 7. One via-point task when the number of iterations is 60.

Figure 7 shows the trajectories of joint space in Swayer. The blue lines represent the joints' trajectories after PI^2-CMA-KCCA learning. The result shows that only the blue lines can pass through the via-point accurately after about 60 iterations.

5.4. Performance Comparison of Four Algorithms

In Table 5, the cost decline rate of PI^2-CMA-KCCA is always higher than that of PI^2, PI^2-CMA, and PI^2-KCCA. When the experimental objects are the same, the new task is more complicated, and the learning performance of PI^2-CMA-KCCA is better than that of the other three algorithms. In addition, especially when the tasks are the same, the greater the number of degrees of freedom of the experimental objects, the better the optimization effect of PI^2-CMA-KCCA than the other algorithms because PI^2-CMA-KCCA not only adjusts the magnitude of exploration noise automatically but also considers the nonlinear heuristic information between the joints.

Table 5. Average decline rate of four algorithms.

Task	PI^2	PI^2-CMA	PI^2-KCCA	PI^2-CMA-KCCA
One Via-Point Task with SCARA	96.5%	97.2%	97.8%	98.5%
Two Via-Point Task with SCARA	84.0%	87.4%	89.7%	92.8%
One Via-Point Task with Swayer	73.0%	81.3%	83.0%	91.5%

6. Conclusions

Compound heuristic information is applied in the paper to guide PI2's variational exploration and expedite the procedure of reinforcement learning. This information is derived by KCCA and CMA together. KCCA infers the nonlinear heuristic information between the joints of robot, and CMA infers the linear heuristic information within single DOF. This information may cause the cost function to drop rapidly with the MLE on the roll-out data. In addition, the proposed algorithm PI^2-CMA-KCCA works like a double closed-loop control system, in which the outer loop discovers and predicts the means of exploring vectors for each DOF and the inner loop discloses and infers the variance of exploring vector for each DOF. In this way, the new algorithm can quickly search the optimal parameters of new tasks. The experimental results on SCARA and Swayer also demonstrate that the algorithm can speed up the process of updating parameters while maintaining the accuracy of completing new tasks, which is suitable for multi-degree-of-freedom objects and more complex tasks.

Author Contributions: Methodology, J.F.; software, C.L.; validation, C.L., X.T.; formal analysis, J.F.; investigation, F.L.; resources, X.T.; data curation, B.L.; writing—original draft preparation, C.L.; writing—review and editing, J.F.; visualization, C.L.; supervision, J.F.; project administration, J.F.; funding acquisition, J.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grants 61773299, 51575412.

Acknowledgments: This work was supported by the National Natural Science Foundation of China under Grants 61773299, 51575412.

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

References

1. Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 987–1008.
2. Takano, W.; Nakamura, Y. Statistical mutual conversion between whole body motion primitives and linguistic sentences for human motions. *Int. J. Robot. Res.* **2015**, *34*, 1314–1328. [CrossRef]

3. Paraschos, A.; Daniel, C.; Peters, J.; Neumann, G. Probabilistic movement primitives. In Proceedings of the 27th Annual Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 2616–2624.
4. Khansari-Zadeh, S.M.; Billard, A. BM: An iterative algorithm to learn stable nonlinear dynamical systems with Gaussian mixture models. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 2381–2388.
5. Ratliff, N.D.; Bagnell, J.A.; Zinkevich, M.A. Maximum margin planning. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 729–736.
6. Sutton, R.S.; Barto, A.G. *Reinforcement Learning*; A Bradford Book; The MIT Press: Cambridge, MA, USA; London, UK, 1998; pp. 665–685.
7. Peters, J.; Schaal, S. Natural Actor-Critic. *Neurocomputing* **2008**, *71*, 1180–1190. [CrossRef]
8. Kober, J.; Peters, J. Learning motor primitives for robotics. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 2112–2118.
9. Peters, J.; Altun, Y. Relative entropy policy search. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; pp. 1607–1612.
10. Theodorou, E.; Buchli, J.; Schaal, S. A Generalized Path Integral Control Approach to Reinforcement Learning. *J. Mach. Learn. Res.* **2010**, *11*, 3137–3181.
11. Stulp, F.; Sigaud, O. Path integral policy improvement with covariance matrix adaptation. In Proceedings of the 29 th International Conference on Machine Learning, Edinburgh, UK, 26 June–1 July 2012; pp. 281–288.
12. Fu, J.; Teng, X.; Cao, C.; Lou, P. Intelligent trajectory planning based on reinforcement learning with KCCA inference for robot. *J. Huazhong Univ. Sci. Technol. (Nat. Sci. Ed.)* **2019**, *47*, 96–102.
13. Ijspeert, A.J.; Nakanishi, J.; Hoffmann, H.; Pastor, P.; Schaal, S. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Comput.* **2013**, *25*, 328–373. [CrossRef] [PubMed]
14. Melzer, T.; Reiter, M.; Bischof, H. Appearance models based on kernel canonical correlation analysis. *Pattern Recognit.* **2003**, *36*, 1961–1971. [CrossRef]