# Robot Intelligent Trajectory Planning based on PCM guided Reinforcement Learning

Xiang Teng[1⋆], Jian Fu[1] Cong Li[1], and ZhaoJie Ju[2]

[1] Wuhan University of Technology, School of Automation, WuHan 430070, China, fujian@whut.edu.cn,
[2] University of Portsmouth, School of Computing, Portsmouth PO1 3HE, London

**Abstract.** Reinforcement Learning (RL) was successfully applied in multi-degree-of-freedoms robot to acquire motor skills, however, it hardly ever consider each joints' relationship, or just think about the linear relationship between them. In order to find the nonlinear relationship between each degrees of freedom (DOFs), we propose a Pseudo Covariance Matrix (PCM) to guide reinforcement learning for motor skill acquisition. Specifically it combined Path Integral Policy Improvement ($PI^2$) with Kernel Canonical Correlation Analysis (KCCA), where KCCA is used to obtain the PCM in high dimensional space and record it as the heuristic information to search an optimal/sub-optimal strategy. The experiments based on robots (SCARA and UR5) demonstrate the new method is feasible and effective.

**Keywords:** Trajectory planning, Learning from demonstration, Kernel Canonical Correlation Analysis, Path Integral Policy Improvement, Pseudo Covariance Matrix

## 1 INTRODUCTION

Reinforcement Learning combined with Demonstration Learning was successfully used in robot to acquire new motor skills. It includes three stages: expression stage, imitation stage and optimization stage, above them the optimization stage is the most important stage to obtain the motor skills, which can realize a reinforcement learning from demonstrate trajectory. The classic methods during this stage include Policy Learning by Weighting Exploration with the Returns (PoWER) [Kober and Peters(2011)], Relative Entropy Policy Search (REPS) [Daniel et al(2016)Daniel, Neumann, Kroemer, and Peters], Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [Gregory et al(2015)Gregory, Martin, and Werner] and $PI^2$ [A. Theodorou et al(2010)A. Theodorou, Buchli, and Schaal]. These methods all update parameters by decreasing the cost function, but $PI^2$ is the most efficient method.

⋆ The author acknowledges the National Natural Science Foundation of China(61773299,515754112), Excellent Dissertation Cultivation Funds of Wuhan University of Technology (2017-YS-066)

PI$^2$ is an intelligent algorithm to avoid the local optimal problem. However its searching strategy is random. Freek Stulp and Olivier Sigaud proposed an algorithm named Path Integral Policy Improvement with Covariance Matrix Adaptation (PI$^2$-CMA) [Stulp and Sigaud(2012)]. It deduces the implicit linear relation among parameters in parameter sapce based on covariance. In this paper, we coupled each joints, and consider the nonlinear relation of parameters not only in its own joint space.

Based on previous research, we use KCCA to get a PCM which can guide the searching strategy. It can infer the nonlinear model among each joints based on experience as the heuristic information, and it can search the optimal/sub-optimal strategy for the new task, we called this method as Path Integral Policy Improvement with Kernel Canonical Correlation (PI$^2$-KCCA).

## 2 Demonstration and Reinforcement Learning based on DMPs-PI$^2$

Dynamical movement primitives (DMPs) is a parametric kinematics model based on Spring-Damping system, which mainly includes conversion system, model system and forcing component [AJ et al(2013)AJ, J, H, P, and S.]. The equation shows in (1). DMPs can achieve Supervised learning and RL by changing its forcing component.

$$\begin{cases} \tau\ddot{x}_t = \underbrace{\alpha_x(\beta_x(g-x_t)-\dot{x}_t)}_{\alpha_z} + \underbrace{\Psi_\theta(s_t)s_t(g-x_0)}_{\alpha_f} \\ \tau\dot{s}_t = -\alpha_s s_t \\ \Psi_\theta(s_t) = \dfrac{\sum_{i=1}^{K}\psi_i\omega_i}{\sum_{i=1}^{K}\psi_i} \end{cases} \tag{1}$$

where $\alpha_z$ represents an ideal Spring-Damping system, $\alpha_f$ represents the forcing component, it denotes the error between ideal acceleration and real acceleration, $\tau$ is the scaling factor of motion duration, $x_t$ is a demonstrated trajectory of one joints, $s_t$ is a phase variable of time which can be described as $s_t = exp(-\frac{\alpha_s t}{\tau})$, $\psi_i$ is the $i$th Gaussian function, $\omega_i$ is the weight of the $i$th Gaussian function, $g$ is the goal position, $x_0$ is the start position and $\alpha_x$,$\beta_x$,$\alpha_s$ is a positive constant.

### 2.1 Learning from Demonstration by LWR

In DMPs model, LWR is an effective way to learn from demonstration. It uses the distance between the query points and sample points as the coefficients of independent variables. LWR is an improved algorithm based on least square

fitting, its cost function is:

$$\begin{cases} J(\theta) = \sum_{j=1}^{n} \sum_{i=1}^{K} \psi_i^{(j)} (y^{(j)} - h_\theta(x^{(j)}))^2 \\ \psi_i^{(j)} = \exp(-\frac{(x^{(j)} - c_i)^2}{2\sigma_i^2}) \end{cases} \tag{2}$$

Here $y^{(j)}$ denotes the $j^{th}$ sample point, $h_\theta(x^{(j)})$ denotes the $j^{th}$ query point, $c_i$ represents the $i^{th}$ center of clustering, $\sigma_i$ denotes the width of $i^{(th)}$ cluster. In order to let the $J(\theta)$ approaches zero, (2) can be converted into (3):

$$y = \frac{\sum_{i=1}^{K} \psi_i \omega_i}{\sum_{i=1}^{K} \psi_i} x \tag{3}$$

Obviously, we can get the $\boldsymbol{\omega} = \{\omega_1, \omega_2, \ldots, \omega_K\}$ from (3). When the parameter of model is confirmed, $\alpha_f$ can be calculated by (1), and then it is easily to get the trajectory by $\alpha_z$ and $\alpha_f$.

### 2.2 Reinforcement Learning by PI$^2$

PI$^2$ uses the Monte Carlo method to spontaneously search the solution which can minimize the cost function in the parameter space [Liu et al(2017)Liu, Qi, Meng, and Fu]. It avoid the curse of dimensionality by its updating strategy, and the estimate of gradient by using probability weighted average. The main principle of PI$^2$ is the first principle of random optimal control based on Hamilton-Jacobi-Bellman (HJB) equation [LIONS(1983)]. In order to get the value function and the optimal control strategy, it convert the target cost function into the path integral by using Feynman-Cutts theorem.

In each step of iteration, we produce 10 trajectories with different cost by add random noise $\boldsymbol{\varepsilon}$ on the parameter $\boldsymbol{\omega}$. The cost function of $k^{th}$ trajectory is:

$$\begin{aligned} S(\tau_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j} dt \; + \frac{1}{2} \times \\ \sum_{j=i}^{N-1} \left(\boldsymbol{\omega} + \boldsymbol{M}_{t_j,k} \boldsymbol{\varepsilon}_{t_j,k}\right)^T \boldsymbol{R} \left(\boldsymbol{\omega} + \boldsymbol{M}_{t_j,k} \boldsymbol{\varepsilon}_{t_j,k}\right) \end{aligned} \tag{4}$$

Where $\Phi_{t_N}$ represents the end cost at time $t_N$, $\boldsymbol{R}$ represents the weight control matrix of square cost function, $\boldsymbol{M}_{t_j,k}$ is the projection matrix of the control matrix on the subspace, and it satisfies the equation: $\lambda \mathbf{R}^{-1} = \boldsymbol{\Sigma}_\varepsilon$, here $\boldsymbol{\Sigma}_\varepsilon$ is the variance of Gaussian noise, $\boldsymbol{\varepsilon_{t_j,k}}$ is the noise of $k^{th}$ trajectory added on $\boldsymbol{\omega}$ in $j^{th}$ time index. $q_{t_j}$ denotes the cost of state in the control system, which is represented by the square of the acceleration, and on the other hand, it can also represent the consumed energy in the system. The probability of the trajectory at time $t_i$ is:

$$P(\tau_i) = \frac{e^{-\frac{1}{\lambda} S(\tau_{t_i})}}{\int e^{-\frac{1}{\lambda} S(\tau_{t_i})} d\tau_i} \tag{5}$$

In (5), the probability is represented by the softmax function, it denotes the discrete probability of each trajectory at time $t_i$, so the probability is inversely proportional to the cost. $\lambda$ is used to control sensitivity of the cost.

$$\delta\omega_{t_i} = \sum_{k=1}^{K} \left[ \boldsymbol{P}\left(\tau_{i,k}\right) \boldsymbol{M}_{t_i,k} \boldsymbol{\varepsilon}_{t_i,k} \right] \tag{6}$$

Equation (6) is be used to update the $\boldsymbol{\varepsilon}$ at time $t_i$. The main idea is to compute the average of weighted noise. The cost is decreasing and converging, because the probability is in inverse ratio to the cost.

## 3 Optimize PI$^2$ by Heuristic Information

### 3.1 Introduction of PCM

According to (5) and (6), the probability of noise added on each joints is equal. The perturbation $\boldsymbol{\epsilon}$ depends on its own joint, so the covariance matrix of $\boldsymbol{\epsilon}$ can be described as:

$$\boldsymbol{\Sigma}_{\epsilon} = \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & 0 \\ & & \ddots & \\ 0 & & & \sigma_M^2 \end{bmatrix} \tag{7}$$

Here, M denotes the number of DOFs, $\Sigma_{\epsilon} \in \mathbb{R}^{M \times M}$ is a symmetric matrix whose $\sigma_1^2 = \cdots = \sigma_M^2 = \sigma^2$

The latest study in cognitive science suggests that the human brain is an organ for statistical analysis and inference. It continually generates hypotheses, and then corrects it based on the sensor. As similar in robot, there is an unknown mode called Heuristic Information between its joints when human given robot a new motor skill. So it would be effective for robot to uses this information to accelerate the learning speed.

Here we consider the nonlinear relationship between each joints as the Heuristic Information. Different from the usual covariance matrix, using kernel method to map $\boldsymbol{\epsilon}$ to $\boldsymbol{\Phi}(\boldsymbol{\epsilon})$ and then (7) converted to (8):

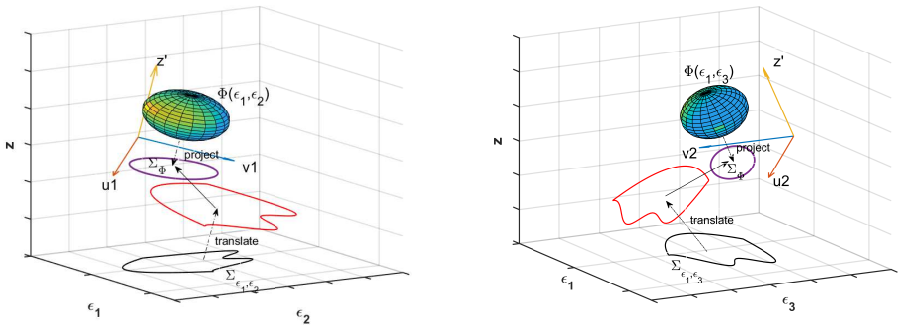
$$\boldsymbol{\Sigma}_{\Phi(\tilde{\boldsymbol{\epsilon}})} = \begin{bmatrix} \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_1) & \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_2) \cdots & \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_M) \\ \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_2, \tilde{\boldsymbol{\epsilon}}_1) & \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_2, \tilde{\boldsymbol{\epsilon}}_2) & \\ \vdots & \ddots & * \\ \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_M, \tilde{\boldsymbol{\epsilon}}_1) & * & \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_M, \tilde{\boldsymbol{\epsilon}}_M) \end{bmatrix} \tag{8}$$

Where $\boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_i, \tilde{\boldsymbol{\epsilon}}_j) = cov(\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i), \boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j))$ is the covariance of $\tilde{\boldsymbol{\epsilon}}_i$ and $\tilde{\boldsymbol{\epsilon}}_j$ on a higher dimensional space. It can represent the heuristic information.

However $\boldsymbol{\Sigma}_{\Phi(\tilde{\boldsymbol{\epsilon}})}$ just express the covariance matrix in the same space $\boldsymbol{\Phi}(\cdot)$. Obviously it is not the best way to do the correlation analysis. We can find a local coordinate system to make a proper projection, and then the correlation

analysis would be efficient. In this paper using Generalized Rayleigh Quotient to find nonlinear correlation between $\tilde{\boldsymbol{\epsilon}_i}$ and $\tilde{\boldsymbol{\epsilon}_j}$. In (8), if $i \neq j$, let $\Theta(\tilde{\boldsymbol{\epsilon}}_i, \tilde{\boldsymbol{\epsilon}}_j) = cov(Proj\Phi(\tilde{\boldsymbol{\epsilon}_i}), Proj\Phi(\tilde{\boldsymbol{\epsilon}_j}))$ as the heuristic information. The perturbation will be guided by covariance $\Theta(\tilde{\boldsymbol{\epsilon}}_i, \tilde{\boldsymbol{\epsilon}}_j)$. By using this method, (8) can change to (9), which can be called as pseudo covariance matrix(PCM). The process is shown in Fig. 1.

$$\boldsymbol{\Sigma}^{+}_{\Phi(\tilde{\boldsymbol{\epsilon}})} = \begin{bmatrix} \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_1) & \boldsymbol{\Theta}(\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_2) \cdots & \boldsymbol{\Theta}(\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_M) \\ \boldsymbol{\Theta}(\tilde{\boldsymbol{\epsilon}}_2, \tilde{\boldsymbol{\epsilon}}_1) & \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_2, \tilde{\boldsymbol{\epsilon}}_2) & \\ \vdots & \ddots & * \\ \boldsymbol{\Theta}(\tilde{\boldsymbol{\epsilon}}_M, \tilde{\boldsymbol{\epsilon}}_1) & * & \boldsymbol{\Gamma}(\tilde{\boldsymbol{\epsilon}}_M, \tilde{\boldsymbol{\epsilon}}_M) \end{bmatrix} \tag{9}$$




(a) (b)

**Fig. 1.** In order to show how it works to change the perturbation's searching strategy, we simply demonstrate the process in $\boldsymbol{\epsilon}_1$, $\boldsymbol{\epsilon}_2$ and $\boldsymbol{\epsilon}_3$. In (a), $\Phi(\epsilon_1, \epsilon_2)$ was project to $[\boldsymbol{u}_1, \boldsymbol{v}_1]$, however, in (b) $\Phi(\epsilon_1, \epsilon_3)$ was project to $[\boldsymbol{u}_2, \boldsymbol{v}_2]$. As a result, we can find the linear relationship under the projection of high dimensional space.

### 3.2 Get the PCM by KCCA

KCCA is an improved algorithm based on Canonical Correlation Analysis [Cai and Huang(2017)], which can get the nonlinear relationship between two sets of data and generate a PCM. In this paper, we using PCM as the heuristic information. After each iteration of PI$^2$, we set the cost decreasing rate as $Trate$. If $Trate$ is greater than its maximum $Tratemax$, we regard this step of searching strategy is useful and record it as PCM by KCCA. Here considering two joints' perturbation $\tilde{\boldsymbol{\epsilon}}_i = \{\boldsymbol{\epsilon}_i^{(1)}, \boldsymbol{\epsilon}_i^{(2)}, \cdots, \boldsymbol{\epsilon}_i^{(n)}\}$ and $\tilde{\boldsymbol{\epsilon}}_j = \{\boldsymbol{\epsilon}_j^{(1)}, \boldsymbol{\epsilon}_j^{(2)}, \cdots, \boldsymbol{\epsilon}_j^{(n)}\}$, where $\tilde{\boldsymbol{\epsilon}}_i, \tilde{\boldsymbol{\epsilon}}_j \in \mathbb{R}^{k \times n}$, $n$ is the number of time samples, and $k$ is the number of $\omega$ in (3), here we set $k$ equals to 10. In order to get the PCM, gauss kernel

method is an effective way to map data to high-dimensional feature space [Cai et al(2016)Cai, Tang, and Wang].

After mapping, CCA is a useful algorithm to find the linear relationship between $\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)$ and $\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j)$. The main principle of CCA is to find two projection vectors $\boldsymbol{\omega}_{\mathrm{i}} \in \mathbb{R}^{N\times 1}$ and $\boldsymbol{\omega}_{\mathrm{j}} \in \mathbb{R}^{N\times 1}$ to maximize the correlation coefficient of $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$, where $\boldsymbol{u}_i = \boldsymbol{\omega}_i^T\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)$ and $\boldsymbol{u}_j = \boldsymbol{\omega}_j^T\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j)$. Since the mean of $\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)$ and $\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j)$ are equal to 0, the mean of $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ are also equal to 0. Then we can obtain the variance of $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ as below:

$$\begin{aligned} var(\boldsymbol{u}_i) &= \frac{1}{m-1}\boldsymbol{\omega}_i^T\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)\boldsymbol{\Phi}^T(\tilde{\boldsymbol{\epsilon}}_i)\boldsymbol{\omega}_i \\ var(\boldsymbol{u}_j) &= \frac{1}{m-1}\boldsymbol{\omega}_j^T\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j)\boldsymbol{\Phi}^T(\tilde{\boldsymbol{\epsilon}}_j)\boldsymbol{\omega}_j \end{aligned} \tag{10}$$

The covariance of $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ is shown as below:

$$cov(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{1}{m-1}\boldsymbol{\omega}_i^T\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)\boldsymbol{\Phi}^T(\tilde{\boldsymbol{\epsilon}}_j)\boldsymbol{\omega}_j \tag{11}$$

[Melzer et al(2003)Melzer, Reiter, and Bischof] propose that the projection vectors $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_j$ should be in the space which is generated by $\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)$ and $\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j)$. So there is $\boldsymbol{\omega}_i = \boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)\boldsymbol{\alpha}$ and $\boldsymbol{\omega}_j = \boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j)\boldsymbol{\beta}$ where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^{n\times 1}$. In (9) $\Theta(\tilde{\boldsymbol{\epsilon}}_i, \tilde{\boldsymbol{\epsilon}}_j)$ can be described as follow:

$$\rho = \frac{\boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta}}{\sqrt{\boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{\alpha}}\sqrt{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta}}} \tag{12}$$

Here we use kernel method to given $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i} = \boldsymbol{\Phi}^T(\tilde{\boldsymbol{\epsilon}}_i)\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_i)$ and $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j} = \boldsymbol{\Phi}^T(\tilde{\boldsymbol{\epsilon}}_j)\boldsymbol{\Phi}(\tilde{\boldsymbol{\epsilon}}_j)$, $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}, \boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j} \in \mathbb{R}^{n\times n}$ are the Gauss Radial Basis Function [Lai and Fyfe(2000)].

From (12), you can obviously find that the value of $\rho$ dose not change with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. So the main problem is to find the appropriate $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to maximize $\boldsymbol{\alpha}^T\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta}$ while $\boldsymbol{\alpha}^T\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{\alpha} = 1$ and $\boldsymbol{\beta}^T\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta} = 1$. The Lagrangian function can be constructed as below:

$$\begin{aligned} L =& \boldsymbol{\alpha}^T\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta} - \lambda_1(\boldsymbol{\alpha}^T\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{\alpha} - 1) - \\ & \lambda_2(\boldsymbol{\beta}^T\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta} - 1) \end{aligned} \tag{13}$$

The derivative of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in (13) is:

$$\begin{aligned} \lambda &= \boldsymbol{\alpha}^T\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta} \\ \boldsymbol{R} \times \begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix} &= \lambda\boldsymbol{D} \times \begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix} \end{aligned} \tag{14}$$

where

$$\boldsymbol{R} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j} \\ \boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i} & \boldsymbol{0}\end{bmatrix} \boldsymbol{D} = \begin{bmatrix}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\end{bmatrix}$$

In order to maximize $\rho$ in (12), $[\boldsymbol{\alpha}^T, \boldsymbol{\beta}^T]^T$ should be the eigenvector corresponding to the maximum eigenvalue of the matrix $\boldsymbol{D}^{-1}\boldsymbol{R}$. Above all, the heuristic information PCM can be obtained.

### 3.3 Predict the Perturbation by Heuristic Information

During the PI$^2$ updating, if $Trate$ is lower than its minimum $Tratemin$. We will user the latest PCM to guide the searching strategy. The first joint's perturbation $\tilde{\boldsymbol{\epsilon}}_1$ is generated randomly as usual, but the other joints' perturbations will calculate by using PCM and $\{(\boldsymbol{\alpha}_{1,2}^T, \boldsymbol{\beta}_{1,2}^T), \cdots, (\boldsymbol{\alpha}_{1,M}^T, \boldsymbol{\beta}_{1,M}^T)\}$ according to the following method.

When all of the sample points are different, $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}$ can be regarded as a full rank matrix [Smola(2008)]. From (14), there is:

$$\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta} = \lambda \boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{\alpha} \tag{15}$$

Because $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}$ is an invertible matrix, we can get an equation about $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}$ and $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}$:

$$\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_j}\boldsymbol{\beta} = \lambda \boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_i}\boldsymbol{\alpha} \tag{16}$$

According to the first joints' perturbation $\tilde{\boldsymbol{\epsilon}}_1$, we can get its kernel space mapping $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_1}$, and calculate $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_2}$ by $(\boldsymbol{\alpha}_{1,2}^T, \boldsymbol{\beta}_{1,2}^T)$ and $\Theta(\tilde{\boldsymbol{\epsilon}}_1, \tilde{\boldsymbol{\epsilon}}_2)$ in PCM, and then deduce the $\tilde{\boldsymbol{\epsilon}}_2$ by $\boldsymbol{K}_{\tilde{\boldsymbol{\epsilon}}_2}$.

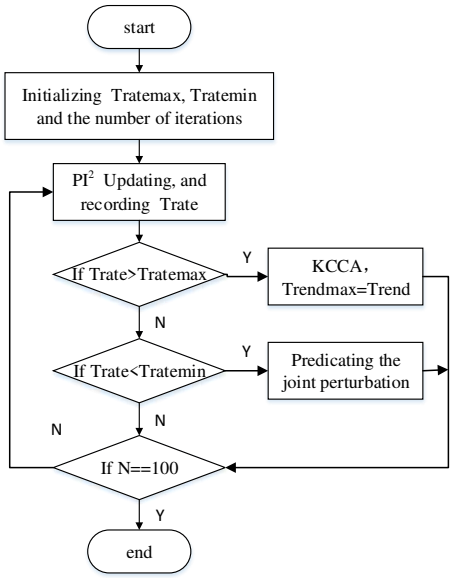



**Fig. 2.** In the flow chart of PI$^2$-KCCA, $Trate$ represent the drop rate of cost. $Tratemax$ represents the upper limit of $Trate$, and $Tratemin$ represents the lower limit of $Trate$.

The whole flow chart is shown in Fig. 2. Here $Tratemax$ is setting to 0.4, and $Tratemin$ is setting to 0.2. When $Tratemin <= Trate <= Tratemax$, the perturbations on each joints are randomly generated. During this flow, it generates hypotheses based on maximum likelihood estimation, and modifies it according to the reward.

# 4 Experiments on SCARA and UR5

## 4.1 Using SCARA via One Point

In this part, we employ SCARA robot arm to show how PI$^2$-KCCA works in a new task. SCARA has three revolute joints $q_1, q_2, q_3$ and one prismatic joint $q_4$. This experiment can be described as five steps:

1. Set point $(20, 0, 0)^{\mathrm{T}}$cm as the start point of the end-effector. In joint space, the start joint vector is $(0, 0, 0)^{\mathrm{T}}$rad.
2. Set point $(4.5, 16, 0)^{\mathrm{T}}$cm as the end point of the end-effector. In joint space, the end joint vector is $(0.7068, 1.1796, 0)^{\mathrm{T}}$rad.
3. Give SCARA a demonstrated trajectory.
4. Acquire SCARA a via-point $(16.4, 11.2, 0)^{\mathrm{T}}$cm at 0.3min by using PI$^2$-CMA and PI$^2$-KCCA.
5. Compare the result of two method.

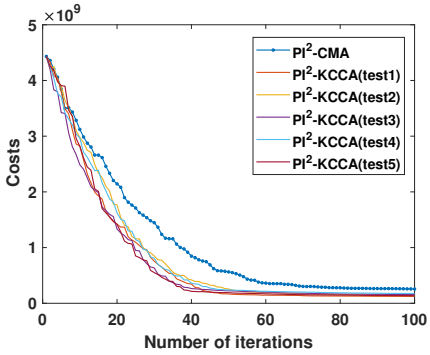



**Fig. 3.** Comparing the cost trend after one experiment by PI$^2$-CMA and five experiments by PI$^2$-KCCA.

In Fig. 3, after 100 times of iteration, we find that PI$^2$-KCCA's drop speed is faster than PI$^2$-CMA. Moreover, the terminal cost of PI$^2$-KCCA is lower. The specific data is shown in Table 1.

Fig. 4 describes joints' trajectories in SCARA's joint space. Here $q_3$ always equals to zero, because $q_3$ is a revolute joints and it does not affect the position of end-effector. It is easily to find that the green line is more accurate than the black dotted line to pass the specific point in time.

In order to show the performance of PI$^2$-KCCA in cartesian space, we use robotics toolbox in matlab to simulate the experiment. The result shows in Fig. 5. The green line represents the joint trajectory under PI$^2$-KCCA, and the black dotted line represents the joint trajectory under PI$^2$-CMA. The end-effector of SCARA should pass though red mark at 0.3min, the black mark denotes the

**Table 1.** Terminal cost comparison of SCARA via one point between PI$^2$-CMA and PI$^2$-KCCA

| experiment | final cost |
|---|---|
| PI$^2$-CMA | 2.561e+08 |
| PI$^2$-KCCA(test1) | 1.229e+08 |
| PI$^2$-KCCA(test2) | 1.452e+08 |
| PI$^2$-KCCA(test3) | 1.556e+08 |
| PI$^2$-KCCA(test4) | 1.758e+08 |
| PI$^2$-KCCA(test5) | 1.373e+08 |

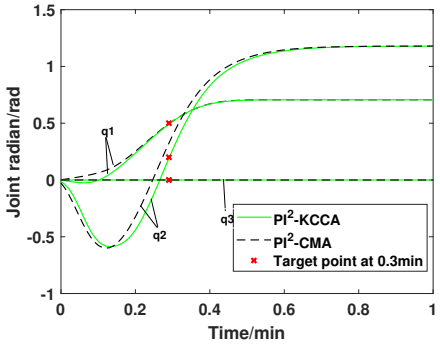



**Fig. 4.** Comparing the trajectories via one point in SCARA's joint space between PI$^2$-CMA and PI$^2$-KCCA.

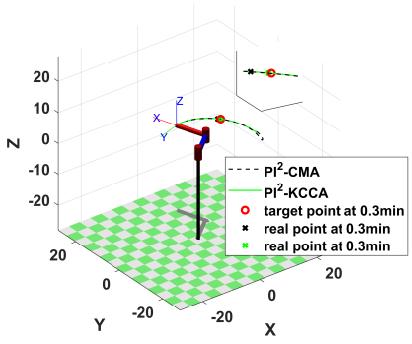



**Fig. 5.** Comparing the trajectories via one point in SCARA's cartesian space between PI$^2$-CMA and PI$^2$-KCCA.

real position at 0.3min under PI$^2$-CMA, and the green mark represents the real position at 0.3min under PI$^2$-KCCA. It is obviously to find that the trajectory under PI$^2$-KCCA is more closer to the red mark at 0.3min.

### 4.2 Using SCARA via Two Points

In this part, we require SCARA to acquire a new motion skill which is more difficult than before. Two via-points $(17.464, 9.541, 0)^{\mathrm{T}}$cm and $(12.834, 13.215, 0)^{\mathrm{T}}$cm are given at 0.15min and 0.23min. The experiment's process is the same as above.

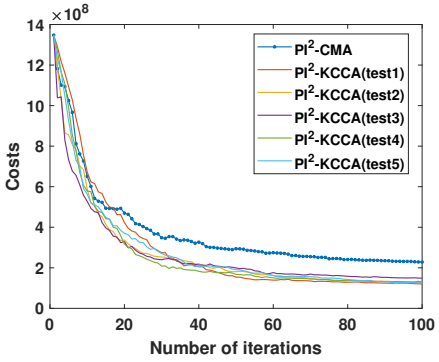



**Fig. 6.** Comparing the cost trend after one experiment by PI$^2$-CMA and five experiments by PI$^2$-KCCA.

**Table 2.** Terminal cost comparison of SCARA via two points between PI$^2$-CMA and PI$^2$-KCCA

| Experiment | Final cost |
|---|---|
| PI$^2$-CMA | 2.284e+08 |
| PI$^2$-KCCA(test1) | 1.207e+08 |
| PI$^2$-KCCA(test2) | 1.297e+08 |
| PI$^2$-KCCA(test3) | 1.491e+08 |
| PI$^2$-KCCA(test4) | 1.296e+08 |
| PI$^2$-KCCA(test5) | 1.269e+08 |

According to Fig. 6 and Table 2, we conclude that the convergence rate in PI$^2$-KCCA is higher than that in PI$^2$-CMA, and under the learning of PI$^2$-KCCA, we can get a much lower terminal cost than PI$^2$-CMA.

Fig. 7 describes the trajectories in joint space, and Fig. 8 describes the trajectories in cartesian space which is simulated by Robotics Tool in Matlab. In Fig. 8, blue point represents the first via-point at 0.15min, and red point represent the second via-point at 0.23min. The subfigure in Fig. 8 shows that the SCARA in PI$^2$-KCCA is more accurate than that in PI$^2$-CMA.

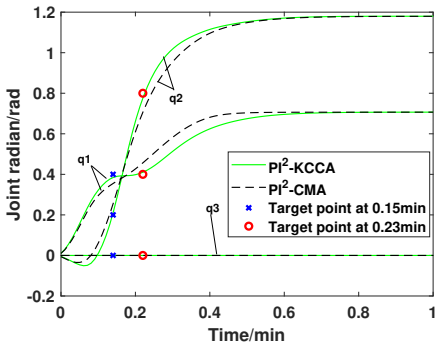



**Fig. 7.** Comparing the trajectories via two points in SCARA's joint space between PI$^2$-CMA and PI$^2$-KCCA.

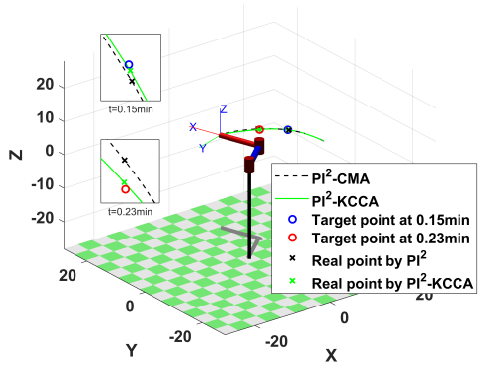



**Fig. 8.** Comparing the trajectories via two points in SCARA's cartesian space between PI$^2$-CMA and PI$^2$-KCCA.

### 4.3 Using UR5 via One Point

In this experiment, we use six DOFs robot UR5 to learn new motor skills. Firstly, we set a start point with $(83.88, -175.09, 601.31)^{\mathrm{T}}$mm, and a terminal point with $(91.23, -630.98, -296.22)^{\mathrm{T}}$mm. Secondly, we give UR5 a demonstrated trajectory from start point to terminal point. Thirdly, we choose a via-point $(-325.92, -552.71, 231.54)^{\mathrm{T}}$mm randomly which stays away from the demonstrated trajectory. In the end, we apply PI$^2$-CMA and PI$^2$-KCCA to UR5 respectively, and compare the results of them.

As shown in Fig. 9, the cost's droop rate of PI$^2$-KCCA is higher than that of PI$^2$-CMA. Table 3 describes the terminal costs of PI$^2$-CMA and PI$^2$-KCCA. We can easily find the terminal cost of PI$^2$-KCCA is smaller than that in PI$^2$-CMA.

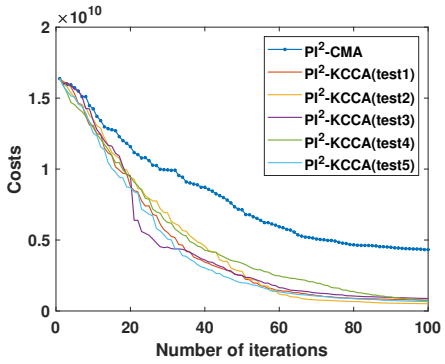



**Fig. 9.** Comparing the cost trend after one experiment by PI$^2$-CMA and five experiments by PI$^2$-KCCA.

**Table 3.** Terminal costs comparison of UR5 via one point between PI$^2$-CMA and PI$^2$-KCCA

| Experiment | Final cost |
|---|---|
| PI$^2$-CMA | 4.326e+09 |
| PI$^2$-KCCA(test1) | 7.627e+08 |
| PI$^2$-KCCA(test2) | 5.127e+08 |
| PI$^2$-KCCA(test3) | 8.790e+08 |
| PI$^2$-KCCA(test4) | 7.310e+08 |
| PI$^2$-KCCA(test5) | 6.591e+08 |

Therefore, the convergence rate of cost under PI$^2$-KCCA is faster than PI$^2$-CMA and PI$^2$-KCCA can get a much lower terminal cost than PI$^2$-CMA.

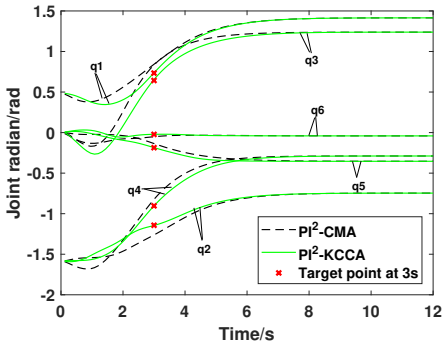
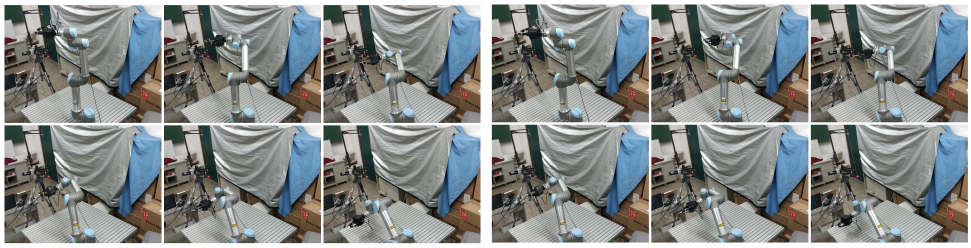



**Fig. 10.** Comparing the trajectories via one point in UR5's joint space between PI$^2$-CMA and PI$^2$-KCCA.

Fig. 10 and Fig. 11 describes the trajectory in joint space and cartesian space respectively. In Fig. 10, just PI$^2$-KCCA can reach the specific position in time in its joint space. However by using PI$^2$-CMA, the joint trajectories can not reach the specific position at the same time. In cartesian space, the end-effector of UR5 can pass through the via-point and touch the red cap under PI$^2$-KCCA learning, but under the learning of PI$^2$-CMA, UR5 fails to search the via-point in its workspace.



(a) Reinforcement Learning by PI$^2$-CMA (b) Reinforcement Learning by PI$^2$-KCCA

**Fig. 11.** Comparing the trajectories via one point in UR5's cartesian space between PI$^2$-CMA and PI$^2$-KCCA.

## 5 Conclusions

**Table 4.** Optimization effect comparison table

| Experiment | Reduction rate(%) | | Optimization(%) |
|---|---|---|---|
| | PI$^2$-CMA | PI$^2$-KCCA | |
| Experiment 1 | 94.2 | 96.9 | 2.7 |
| Experiment 2 | 83.0 | 90.5 | 7.5 |
| Experiment 3 | 73.6 | 96.0 | 22.4 |

According to Table 4, the average cost reduction rate of PI$^2$-KCCA is always higher than PI$^2$-CMA. When the experimental objects are the same, the more complex the new task target is, the higher optimization will be, because of the heuristic exploration of PI$^2$-KCCA. When the tasks are the same, the more DOFs the objects have, the higher optimization of PI$^2$-KCCA will be, because PI$^2$-KCCA considers the nonlinear correlation between each joints.

Recently RL has received strong attention in the field of intelligent robots. Accelerating the iteration speed of RL is important. In this paper, we propose a novel algorithm PI$^2$-KCCA base on PI$^2$-CMA to find the heuristic information

as a PCM during the convergence of cost. KCCA is an effective way to establish the nolinear relationship between each joints, and we use it to recode the relationship as the heuristic information while the convergence rate is greater than the threshold, to learn a appropriate perturbation strategy, and apply this strategy to predict joints' noise when the convergence rate is going down. According to the experiments, PI$^2$-KCCA can not only speed up the convergence rate, but improve the accuracy for new tasks.

## References

[A. Theodorou et al(2010)A. Theodorou, Buchli, and Schaal] A Theodorou E, Buchli J, Schaal S (2010) A generalized path integral control approach to reinforcement learning. Journal of Machine Learning Research pp 3137–3181

[AJ et al(2013)AJ, J, H, P, and S.] AJ I, J N, H H, P P, S S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. Neural Computation (No.2):328–373

[Cai and Huang(2017)] Cai JCJ, Huang XHX (2017) Robust kernel canonical correlation analysis with applications to information retrieval. Engineering Applications of Artificial Intelligence pp 33–42

[Cai et al(2016)Cai, Tang, and Wang] Cai JCJ, Tang YTY, Wang JWJ (2016) Kernel canonical correlation analysis via gradient descent. Neurocomputing pp 322–331

[Daniel et al(2016)Daniel, Neumann, Kroemer, and Peters] Daniel C, Neumann G, Kroemer O, Peters J (2016) Hierarchical relative entropy policy search. JOURNAL OF MACHINE LEARNING RESEARCH

[Gregory et al(2015)Gregory, Martin, and Werner] Gregory MD, Martin SV, Werner DH (2015) Improved electromagnetics optimization: The covariance matrix adaptation evolutionary strategy. IEEE Antennas and Propagation Magazine (No.3):48–59

[Kober and Peters(2011)] Kober JJKT, Peters JJPT (2011) Policy search for motor primitives in robotics. Machine Learning (No.1-2):171–203

[Lai and Fyfe(2000)] Lai PL, Fyfe C (2000) Kernel and nonlinear canonical correlation analysis. In: IEEE/INNS/ENNS International Joint Conference on Neural Networks (IJCNN 2000), COMO, ITALY, IEEE/INNS/ENNS International Joint Conference on Neural Networks (IJCNN 2000)

[LIONS(1983)] LIONS PL (1983) Optimal-control of diffusion-processes and hamilton-jacobi-bellman equations .1. the dynamic-programming principle and applications. COMMUNICATIONS IN PARTIAL DIFFERENTIAL EQUATIONS (No.10):1101–1174

[Liu et al(2017)Liu, Qi, Meng, and Fu] Liu JLJ, Qi YQY, Meng ZMZY, Fu LFL (2017) Self-learning monte carlo method. PHYSICAL REVIEW B (No.4)

[Melzer et al(2003)Melzer, Reiter, and Bischof] Melzer TMPT, Reiter M, Bischof H (2003) Appearance models based on kernel canonical correlation analysis. Pattern Recognition: The Journal of the Pattern Recognition Society (NO.9):1961–1971

[Smola(2008)] Smola AJ (2008) Learning with kernels | support vector machines. Lecture Notes in Computer Science ,2008 ,42 (4) :1-28

[Stulp and Sigaud(2012)] Stulp F, Sigaud O (2012) Path integral policy improvement with covariance matrix adaptation. Computer Science