Humanoid Robot Motion Modeling Based on Time-Series Data Using Kernel PCA and Gaussian Process Dynamical Models

Paper:

Humanoid robot motion modeling based on time-series data using kernel PCA and Gaussian Process Dynamical Models

Jian Mi and Yasutake Takahashi

Department of Human and Artificial Intelligent Systems,
Graduate School of Engineering, University of Fukui,
3-9-1, Bunkyo, Fukui, Fukui, 910-8507, Japan;
E-mail: {jmi, yasutake}@ir.his.u-fukui.ac.jp

[Received 00/00/00; accepted 00/00/00]

In this article, contrary to the popular studies on human motion learning, we focus on addressing the problem of humanoid robot motions directly. Performances of different kernel functions with principal components analysis (PCA) in Gaussian process dynamical models (GPDM) are investigated to build efficient humanoid robot motion models. A novel kernel-PCA-GPDM method is proposed for building different types of humanoid robot motion models. Compared with standard-PCA-GPDM method auto-encoder-GPDM method, our proposed method is more efficient in humanoid robot motion modeling. In this work, three types of NAO robot motion models are studied: walk-model, lateral-walk model, and wave-hand model where motion data are collected from an Aldebaran NAO robot using Magnetic Rotary Encoders sensors. Using kernel-PCA-GPDM method, the motion data are firstly projected from the 23 high dimension observation space to a 3 dimension low latent space. Then, three types humanoid robot motion models are learned in the 3D latent space. Compared with other kernel-PCA-GPDM or auto-encoder-GPDM, our proposed novel kernel-PCA-GPDM method performs efficiently in the motion learning. Finally, we realize humanoid robot motion representation to verify the motion models that we build. The experimental results show that our proposed kernel-PCA-GPDM method builds efficient and smooth motion models.

Keywords: Humanoid robot; dimensionality reduction; kernel PCA; humanoid motion models; Gaussian Process Dynamical Models.

1. Introduction

Motion learning has become popular in the field of machine learning. Studies of human motion learning and imitation systems[1][2][3] progress in the development of humanoid robots motion learning in recent years. Various machine learning methods are proposed to make a humanoid robot behave and decide like human beings[4][5]. Among those motion learning studies, most researchers focused on human motion learning [6][7][8] or imitation systems of human motion using humanoid robots[9][10]. Most conventional studies on learning human motions apply optimized human motion data into a humanoid robot through the studying of human motion models, such as imitation system[11]. In our research, we focus on studying inherent motion characteristics of a humanoid robot and constructing humanoid robot motion models directly.

The humanoid robot motion models are necessary for real-time imitation of the humanoid robot to reduce the huge exploration space. A humanoid robot has about 20 joints as a whole body and the 20 DoF space becomes the exploration space and too huge to find a set of good parameter values for imitation in real-time. The previous studies [6][8] revealed that the human motion can be represented in a few DoF space even though a human has huge amount of degree of freedoms to generate his/her motions. The other studies [9][12] also showed that the humanoid motion can be represented in a few DoF space as well. They suggest that humanoid motion models with a few DoF space can be used to explore parameters efficiently for the real-time imitation of the humanoid robot.

Yo Kondo and Yasutake Takahashi[9] introduced the idea above for the real-time imitation of the humanoid robot. They built an imitation system and adopted auto-encoder to build humanoid robot motion models. They greatly reduce the computational cost in imitation by using 3D humanoid robot motion models. However, the learning system they built is complex. They divided full robot body joints into four parts, two arms and two legs where they built hierarchical seven auto-encoders to train the motion data separately. The seven auto-encoders needs relatively long learning time and a lot of training data. The motion trajectories in the learned space are unstable and vague.

In this paper, we use machine learning methods named kernel PCA and Gaussian Process Dynamical Models(GPDM) addressing humanoid robot motion learning. GPDM is very efficient for high dimensional data training and motion learning[7, 13–15]. Unlike conventional deep learning methods which are also powerful and widely applied for motion learning, GPDM does not need huge amounts of training data. It can train a motion model effi-

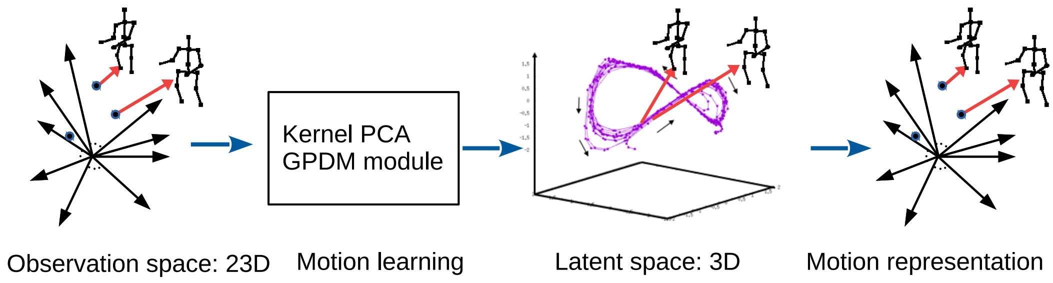


Fig. 1. (a) The left figure show the NAO robot motion data in 23D observation space. Each pose of a NAO robot is defined with 23 motion features. (b)Through the motion learning with kernel-PCA-GPDM method, high dimensional motion features are extracted into a low 3D latent space. The features are learned in the low 3D latent space. The right side figure of kernel-PCA-GPDM module shows a motion model learned with kernel-PCA-GPDM method. (c) After a motion model is built, we restore the learned 3D motion features back to 23 dimension space. The motion model is verified by humanoid robot motion representation

ciently with modest amounts of training data and the time cost is also much less. As shown in **Fig. 1**, the kernel-PCA-GPDM method first projects 23 dimensional motion data into 3D latent space. Then, the 3D motion feature data are learned and motion models can be built in the 3D latent space. Finally, we restore the learned 3D motion data back to 23 dimension space. The motion model is verified by means of motion representation.

Generally, the curse of dimensionality is a serious problem in motion learning. It is hard to learn high dimensional data directly as the computational cost increases greatly. In motion learning, the high dimensional data is usually projected into a low state space. Raquel Urtasun et al.[14] used a similar dimensionality reduction method as Neil D. Lawernce[16] which turns out to be equivalent to PCA. Jian Mi and Yasutake[17] also used PCA for dimensionality reduction in their method. However, the humanoid robot motion models they built is not smooth. Taehwan Kim et al.[8], Yo Kondo and Yasutake Takahashi[9] both adopted auto-encoder for dimensionality reduction. Other methods, such as Locally linear embedding(LLE) and Isomap, are not suitable for humanoid robot motion data because they assume the observed data are densely sampled on manifold[6]. In this paper, we use kernel PCA[18] replaced the standard-PCA method in GPDM for dimensionality reduction. The kernel PCA has advantages over the other methods such as it addresses linear/non-linear data efficiently with different kernels and we build smooth humanoid robot motion models using kernel-PCA-GPDM method. In this paper,

1. We build motion models of a humanoid robot instead of focusing on human motion models.
2. The full body joints data of a humanoid robot is learned with kernel-PCA-GPDM rather than dividing the full robot body joints data into several parts.
3. We investigate performances of different kernel functions to figure out which kernel-PCA-GPDM executes better humanoid robot motion modeling. In details, as shown in **Fig. 1**, we reduce the motion data of a humanoid robot from 23 high dimension observation space into a 3 dimension latent space and then, motion models are learned with kernel-PCA-GPDM method in the 3D low latent space.
4. A novel RBF-Hyperbolic Tangent(RBF-HT) kernel-PCA-GPDM method is proposed for motion learning.
5. Three types of humanoid robot motion models are learned with kernel-PCA-GPDM and we also compare our proposed RBF-HT kernel-GPDM method with auto-encoder-GPDM and standard-PCA-GPDM methods. The results show the efficiency of our proposed method.
6. The motion model is verified through the motion representation by restoring the learned 3D motion data.

2. Joints definition of a NAO humanoid robot

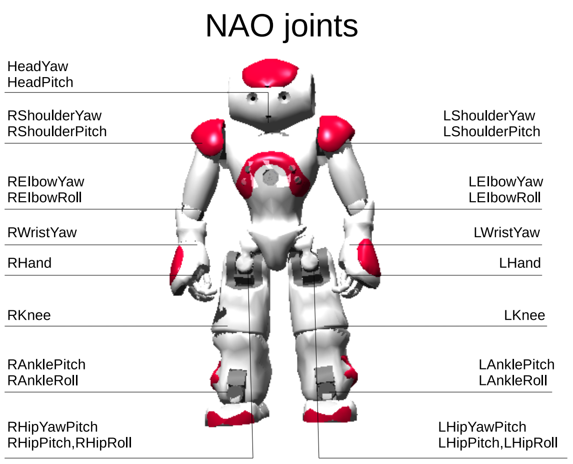


Fig. 2. 25 joints definitions of a NAO robot

In our research, we use an Aldebaran NAO robot shown in **Fig. 2**. The Aldebaran NAO robot is very famous among humanoid robots. It has 25 DOF joints and can perform many tasks. Kinds of research are based on NAO platform because it is very powerful for programming. Rodrigue et al.[19] used a NAO robot as a home rehabilitation assistant. Chao Li et al.[20] performed visual localization and object tracking with a NAO robot in dynamic environment. Yongsheng et al.[1], in their work, a Kinect was used with a NAO robot to realize real-time full body human imitation.

The definition of 25 joints are indicated in **Fig. 2**. There are 26 joints shown in **Fig. 2**. In fact, the LHipYawPitch joint and RHipYawPitch joint share the same motor and cannot be controlled separately. We consider LHipYawPitch joint and RHipYawPitch joint as one joint. The sensors that a NAO robot uses for collecting data are Magnetic Rotary Encoders (MRE). The MRE sensors are embedded in the joints of a NAO robot. In this way, the joint angles data can be obtained when a NAO robot moves.

3. Related Work

A lot of works address the problem of human motion imitation and building human motion models [21][22][23]. Here, we pay attention to humanoid robot motion models. We briefly introduce several previous research related to our work.

Yongsheng Ou et al[1] used a Microsoft Kinect sensor for real-time human motion imitation with a NAO robot. They solved inverse kinematics by means of Levenberg-Marquart optimization process for imitation. They focused on mimicking human motions. Similarly, Haiyu Zhu et al[24] provided a method using one RGB camera for the purpose of building personalized parametric models of a dynamic human body. Judith Bütepage et al[25] developed an unsupervised representation learning scheme for long-term prediction of human motion. They addressed the problems of representation learning for human motion prediction and classification. Our research is different from those research, we focus on the field of studying humanoid robot motions instead of learning human motion models nor the imitation systems. In order to investigate the characteristic of humanoid robots movements, it is necessary to study the humanoid robot motions and build humanoid robot motion models directly rather than human motion models.

Auto-encoder, a common machine learning method, is widely used in dimensionality reduction and training data[8][26][27]. Taehwan Kim et al[8] used auto-encoder-GPDM method for human motion learning. They reduced accelerometer and gyro sensor data from 16D to 2D. Unlike their research, we address humanoid robot joints data. Yo Kondo and Yasutake Takahashi[9], also addressed humanoid robot joints data. They used an auto-encoder method for dimensionality reduction. In their research, they used auto-encoder to reduce high dimensional humanoid robot joints data from the 18 dimension space to a 3D latent space. For details, they divided the NAO robot body joints into 4 parts, two arms and two legs. The four parts were processed with auto-encoder separately. In the whole process, they used seven networks and reduced 18 dimensional robot joints data into a 3D space. In their research, complex networks are adopted to train amounts number of motion data in 3D space. Correspondingly, the complex networks need much more time to train motion data. On the contrary, we want to process the 23 dimensional robots joints data directly with kernel-PCA-GPDM rather than divide the robots joints data into several parts. In the meantime, we want to cut down the computational cost of training robots motion data to build an efficient humanoid robot motion model with small amount data.

Gaussian process dynamical model is an efficient machine learning method for motion data training. Jack M. Wang et al[6], used Gaussian Process Dynamical Model to address high dimensional human pose and motion data on CMU data base. In their research, they reduced the high dimensional data from 50 DOFs to 3 DOFs with GPDM and their method learned an effective representation of nonlinear dynamics in the latent space. Raquel Urtasunet al[28], also used GPDM method for learning human pose and motion priors for 3D people tracking. Unlike those researches, we are aiming at training humanoid robot motion data and building efficient humanoid robot motion models in a low latent space. In [17], they used PCA in their GPDM method. However, PCA is not suitable for humanoid robot motion data because the motion data is nonlinear. In this paper, we propose a kernel-PCA-GPDM method to build humanoid robot motion models with small amount of humanoid robot motion data. Particularly, a novel RBF-HT kernel is proposed to build smooth motion models. We can predict the next states of a humanoid robot with the learned motion models. It is helpful in humanoid robots motion design, humanoid robots control and other humanoid robots fields, such as humanoid rescue robots, humanoid rehabilitation robots, etc..

4. Data Process using kernel PCA Gaussian Process Dynamical Models

Gaussian Process Dynamical Model(GPDM) is a latent variable dynamical model[6]. Typically, GPDM is nonlinear, efficient for motion model building with a small amount of motion data. Besides, GPDM is also powerful in training time series data[29]. In this section, we briefly introduce the kernel-PCA-GPDM method.

4.1. Dimensionality reduction: Principal Components Analysis

Dimensionality reduction plays an important part of GPDM. The extracted motion features affect the data training greatly. Good extracted motion features help to build good motion models. It is important to figure out a suitable way to realize proper dimensionality reduction. The principal components analysis(PCA)[30] is widely

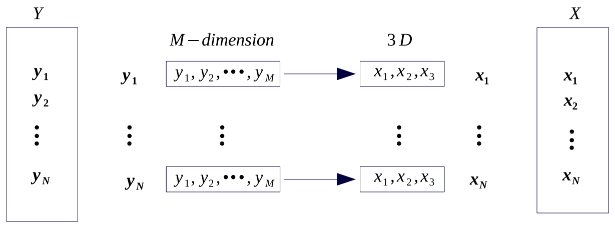


Fig. 3. Dimensionality reduction with PCA: $MD \rightarrow 3D$

used in feature extraction and dimensionality reduction. **Figure 3** shows a PCA dimensionality reduction model where high dimensional humanoid robot motion data is projected from the M dimension observation space $\boldsymbol{y}$ to a 3D low latent space $\boldsymbol{x}$.

A vector in the observation space $\boldsymbol{y}$ corresponds to a pose configuration of a humanoid robot. Sequences of poses define a motion trajectory. In **Fig. 3**, $Y = [\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_N}]^T$, is the data set of observation vector. N is the number of elements that Y contains. Each element Y_n represents a pose of the motion trajectory. $\boldsymbol{y_n} = [y_1, y_2, \cdots, y_M]^T$, here, M is the dimension of pose $\boldsymbol{y_n}$. $X = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}]^T$, is the latent variable addressed by PCA. The high dimensional motion data is reduced from MD to 3D which is shown in **Fig. 3**. Each $\boldsymbol{x_n} = [x_1, x_2, x_3]^T$ contains a 3D vector. **Algorithm 1** shows the procedure of standard PCA reduce high dimensional data to d dimension[17]. The standard PCA performs dimensionality reduction directly from high dimension space into low dimension space. However, it addresses linear transformation. It cannot perform good feature extraction on nonlinear motion data.

Algorithm 1 Principal Components Analysis: reduce high dimensional data to d dimension

```
1: Input data set $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N]^T$, output data set $X$;
2: $Y \leftarrow Normalization$ on $Y$;
3: Calculate covariance matrix: $\sum_{cov}$;
4: Calculate eigenvectors $\boldsymbol{U}$ and eigenvalues $\boldsymbol{\lambda}$;
5: Reshape $\{\boldsymbol{\lambda}, \boldsymbol{U}\}$ to $\{\lambda_i, U_i\}$, where $1 \le i \le N$ and
   $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_N$;
6: $X = Y\boldsymbol{U}_{1:d}$ where $d < N$.
```

4.2. Kernel Principal Components Analysis

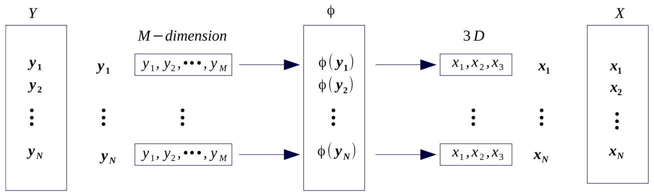


Fig. 4. Dimensionality reduction with kernel PCA

Instead, we use kernel principal components analysis for nonlinear humanoid robot motion data. Unlike PCA, kernel PCA addresses nonlinear data. It first maps the input data $\boldsymbol{y_n}$ into a much higher dimensional Hilbert space $\boldsymbol{\phi(y_n)}$ as shown in **Fig. 4**. The dimension of the Hilbert space $\boldsymbol{\phi(y_n)}$: $D_{\phi(y_n)} \rightarrow \infty$. The kernel PCA maps data from the Hilbert space to a low space instead of the original space. However, it is difficult to obtain the nonlinear mapping $\phi(\boldsymbol{y}_n)$. To solve this problem, a kernel function $k(\phi_i, \phi_j)$ is adopted in kernel PCA[18]. In this way, we transform the original problem $\phi(\boldsymbol{y}_n)$ into kernel computing. The kernel PCA is shown in **Algorithm 2**. In kernel PCA, kinds of kernel functions are used such as linear kernel function, Gaussian kernel function, Hyperbolic Tangent(Sigmoid) kernel function, *etc.*. The linear kernel function is defined in **Eq. (1)**. It is the simplest kernel function and defined by inner product $< y, y' >$. **Equation (2)** shows a polynomial kernel function. It is defined by three parameters, ξ, a constant c and the polynomial degree d. The Hyperbolic Tangent(Sigmoid) kernel function is also known as Multilayer Perceptron(MLP) kernel is defined in **Eq. (3)**. It is popular as activation function in neural networks. Two parameters are in this kernel function, ξ and c. The Gaussian kernel function is shown as **Eq. (4)**. It is also known as Radial basis function(RBF) kernel and is widely used in machine learning field. The parameter σ plays the major role which should be estimated according to the training data. If σ is overestimated, the high dimensional projection will lose its nonlinear power. Otherwise, if σ is underestimated the function will lack regularization and it will be sensitive to noises in the training data. It is very important to figure out an appropriate σ. Those are the four popular kernels in machine learning. In this paper, we also propose a novel kernel named RBF-HT kernel by combining the RBF kernel and Hyperbolic Tangent kernel. The details of the novel kernel is described in **Section 5.4** .

Algorithm 2 Kernel PCA: map high dimensional data into d dimension

```
1: Input data set $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N]^T$, output data set $X$;
2: Select a kernel $k(\phi_i, \phi_j)$, where $\phi_i, \phi_j \in \phi(\boldsymbol{y}_n)$;
3: Construct and centralize kernel matrix $K = [k(\phi_i, \phi_j)]$:
   $\tilde{K} = centralize(K)$[31];
4: Address eigenvalue problem $\tilde{K}\alpha_i = \lambda_i \alpha_i$, where $\alpha_i$ is
   coefficient;
5: Each output $\boldsymbol{x}_i = \sum_{i=1}^{N} \alpha_{ji} K(\boldsymbol{y}, \boldsymbol{y}_i)$.
```

$$k(y, y') = y^T y' \quad \ldots \ldots \ldots \ldots \quad (1)$$

$$k(y, y') = (\xi y^T y' + c)^e \quad \ldots \ldots \ldots \quad (2)$$

$$k(y, y') = tanh(\xi y^T y' + c) \quad \ldots \ldots \ldots \quad (3)$$

$$k(y, y') = exp\left(-\frac{1}{2\sigma^2}\left\|y - y'\right\|^2\right) \quad \ldots \ldots \quad (4)$$

4.3. Dynamical Models

In our work, the humanoid robot motion data are pre-processed with kernel PCA reducing 23 high dimensional data into 3D. Then, the motion data are learned in the 3D latent space. In Gaussian Process Dynamical model, the learning parameters are $\{Y^T, X^T, \bar{\alpha}, \bar{\beta}, \omega\}$ where Y^T, X^T are the observation and corresponding latent variable data sets, $\bar{\alpha}, \bar{\beta}$ and ω are the learning hyper-parameters and scale parameter, respectively. The state space models for the dynamical systems are defined as follows:

$$\boldsymbol{x}_t = f_x(\boldsymbol{x}_{t-1}) + N(0, \sigma_x^2 I) \quad \dots\dots\dots \quad (5)$$

$$\boldsymbol{y}_t = f_y(\boldsymbol{x}_t) + N(0, \sigma_y^2 I) \quad \dots\dots\dots \quad (6)$$

where, $\boldsymbol{x}_t$ is the d dimensional latent coordinates at time t and $\boldsymbol{y}_t$ is the observation pose vector. $N(0, \sigma_x^2 I), N(0, \sigma_y^2 I)$ are white Gaussian noise processes. f_x, f_y are combination of nonlinear basis functions $\varphi(x), \psi(x)$ with weights A and B. Consequently, **Eq. (5)** and **Eq. (6)** become

$$\boldsymbol{x}_t = \sum_i a_i \varphi_i(\boldsymbol{x}_{t-1}) + N(0, \sigma_x^2 I) \quad \dots\dots \quad (7)$$

$$\boldsymbol{y}_t = \sum_j a_i \psi_j(\boldsymbol{x}_t) + N(0, \sigma_y^2 I), \quad \dots\dots\dots \quad (8)$$

here, $A = [a_1, a_2, \cdots]$, $B = [b_1, b_2, \cdots]$. In regression, the number of basis functions should be selected first to fit the model. Then, the parameters A and B should be estimated. Note that, parameters A and B are nuisance from the view of Bayesian perspective, so that it is necessary to marginalize over the parameters. Parameter A can be marginalized out as [6] performed. Parameter B can be marginalized out in a similar way as [32][33] to obtain the probability of the training data which is shown in **Eq. (9)**.

$$p(Y|X, \overline{\beta}, \omega) = \frac{|\omega|^N}{\sqrt{(2\pi)^{NM} |K_Y^M|}} \exp\left(-\frac{1}{2} tr(K_Y^{-1} Y \omega^2 Y^T)\right) \quad .. \quad (9)$$

A kernel matrix K_Y is defined by $(K_Y)_{ij} = k_Y(x_i, x_j)$ In this research, the RBF kernel is adopted to the GPDM, defined as follow,

$$k_Y(x, x') = \beta_1 \exp\left(-\frac{1}{2\beta_2^2} \left\| x - x' \right\|^2\right) + \beta_3 \delta_{x,x'}. \quad (10)$$

x and x' are latent variables in $\boldsymbol{x}$. β_1 is the scale parameter. β_2 is the standard deviation that controls the kernel width and β_3 represents the variance of the noise in **Eq. (6)**.

Similarly, the probability of the latent variable X is calculated by

$$p(X|\overline{\alpha}) = \frac{p(\boldsymbol{x}_1)}{\sqrt{(2\pi)^{(N-1)d} |K_X|^d}} \exp\left(-\frac{1}{2} tr(K_X^{-1} X_o X_o{}^T)\right), \quad \dots\dots \quad (11)$$

where d is the latent space dimension (in our research, set as 3). $X_o = [\boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_N]$, and kernel matrix K_X is $(N-1) \times (N-1)$ which is generated from data set $[\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{N-1}]$. In addition, $\boldsymbol{x}_1$ is assumed to have an isotropic Gaussian prior. A combination kernel of linear and RBF is adopted.

$$k_X(x, x') = \alpha_1 \exp\left(-\frac{1}{2\alpha_2^2} \left\| x - x' \right\|^2\right) + \alpha_3 x^T x' + \alpha_4 \delta_{x,x'} \quad \dots\dots \quad (12)$$

where x and x' are latent variables in $\boldsymbol{x}_{1:N-1}$. α_1 and α_2 are parameters of the part of RBF kernel while α_3 is the parameter of the linear kernel part. In details, α_1 is the scale parameter of RBF kernel, and α_2 is the standard deviation that controls the RBF kernel width; α_3 is the scale parameter of the linear kernel; α_4 is the variance of the noise in **Eq. (5)**. The advantage of the combination kernel is that the linear part can work efficiently when the predictions are far from original data.

The hyper-parameters are estimated through GPDM learning. The motion models are built by training the input pose data. We take a similar way as [6] performed to apply simple prior distributions over the kernel hyper-parameters. $p(\bar{\alpha}) \propto \prod_i \alpha_i^{-1}$, $p(\bar{\beta}) \propto \prod_i \beta_i^{-1}$ and $p(\omega) = \prod_{m=1}^{M} \frac{2}{k\sqrt{2\pi}} \exp(-\frac{\omega_m^2}{2k^2})$. Take all the parameters into consideration, the model defined by GPDM along time series data is

$$p(X, Y, \overline{\alpha}, \overline{\beta}, \omega) = p(Y|X, \overline{\beta}, \omega) p(X|\bar{\alpha}) p(\bar{\alpha}) p(\bar{\beta}) p(\omega). \quad (13)$$

In the process of GPDM, the observation data Y, as the training data, is known. The latent variable X is first initialized by kernel PCA as shown in **Algorithm 2**. Through the learning of Y, the unknown parameters $\{X, \bar{\alpha}, \bar{\beta}, \omega\}$ are estimated.

$$\mathscr{L} = \frac{M}{2} ln|K_Y| + \frac{1}{2} tr(K_Y^{-1} Y W^2 Y^T) - N ln|\omega| + \frac{d}{2} ln|K_X| + \frac{1}{2} tr(K_X^{-1} X_o X_o^T) + \sum_i ln\alpha_i + \sum_i ln\beta_i + \gamma \quad . \quad (14)$$

Here, γ is a constant. According to **Eqs. (9)** and **(11)**, we can obtain

$$log(p(Y|X, \overline{\beta}, \omega)) = log(\frac{|\omega|^N}{\sqrt{(2\pi)^{NM} |K_Y^M|}} \exp\left(-\frac{1}{2} tr(K_Y^{-1} Y \omega^2 Y^T)\right)), \quad .. \quad (15)$$

$$log(p(X|\overline{\alpha})) = log(\frac{p(\boldsymbol{x}_1)}{\sqrt{(2\pi)^{(N-1)d} |K_X|^d}} \exp\left(-\frac{1}{2} tr(K_X^{-1} X_o X_o{}^T)\right)). \quad .. \quad (16)$$

With **Eqs. (15)** and **(16)**, the hyper-parameters are estimated by minimizing the negative log posterior of **Eq.**

(14). ω is estimated by minimizing $\mathscr{L}$(**Eq. (14)**) with respect to $\{X, \bar{\alpha}, \bar{\beta}\}$. A scaled conjugate gradient method is applied and looped for L times as shown in line 9, **Algorithm 3**. As shown in **Algorithm 3**, the GPDM learns $I_{teration}$ times in the whole process.

Algorithm 3 GPDM

```
 1: Input data $Y$, output parameters $\{X, \overline{\alpha}, \overline{\beta}, \mathrm{W}\}$
 2: Initialize $X$ with kernel PCA based on $Y$ with $d$ dimensions, as **Algorithm 1**
 3: Initialize $\overline{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, $\overline{\beta} = (\beta_1, \beta_2, \beta_3)$, $\omega_j$
 4: for $i = 1$ to $I_{teration}$ do
 5:     for $j = 1$ to $M$ do
 6:         $\omega$: estimated by minimizing $\mathscr{L}$(Equation (14))
 7:     end for
 8:     for $l = 1$ to $L$ do
 9:         $\{X, \bar{\alpha}, \bar{\beta}\}$: estimated by optimizing $\mathscr{L} \propto \{X, \bar{\alpha}, \bar{\beta}\}$
10:     end for
11: end for
```

5. Motion models of a NAO Robot building with kernel-PCA-GPDM method

In this paper, three motion models are learned in 3D latent space using kernel-PCA-GPDM method. For the NAO robot, we use software Choregraphe and Webots to design and simulate motion models. The three motion models we build are models of walk-forward, lateral-walk and hand-wave. **Figure 5(a)** shows the walk-forward model of a NAO robot. **Figure 5(b)** shows the lateral-walk model of a NAO robot. In **Fig. 5(c)**, the NAO robot waves a single hand. The NAO robot acts along time t which is shown in **Fig. 5**. We collect time series pose data of a NAO robot and each pose is defined by 25 joints data. In the experiments, we collect 1000 frames source data for each motion. 500 frames of each motion data are selected to form the training data set. The parameters in **Algorithm 3** are set as $I_{teration} = 50$, $L = 10$.

5.1. Motion models built with standard-PCA-GPDM method

We first address time series motion data with standard-PCA-GPDM method. Here, we choose the source data of walk-forward to calculate the cumulative contribution rate. The statistic data of cumulative contribution rate under different dimensions is shown in **Table 1**. Generally, the cumulative contribution rate should be larger than 85% because if the cumulative contribution rate is less than 85%, which means much data loss and the data cannot represent the original data well. This is the reason we reduce the high dimension data into 3D latent space instead of 2D latent space. The motion models learned by standard-PCA-GPDM method are shown in **Fig. 6**. 23 dimensional motion data are projected into a 3D latent space. The 3D latent space is defined by three axes

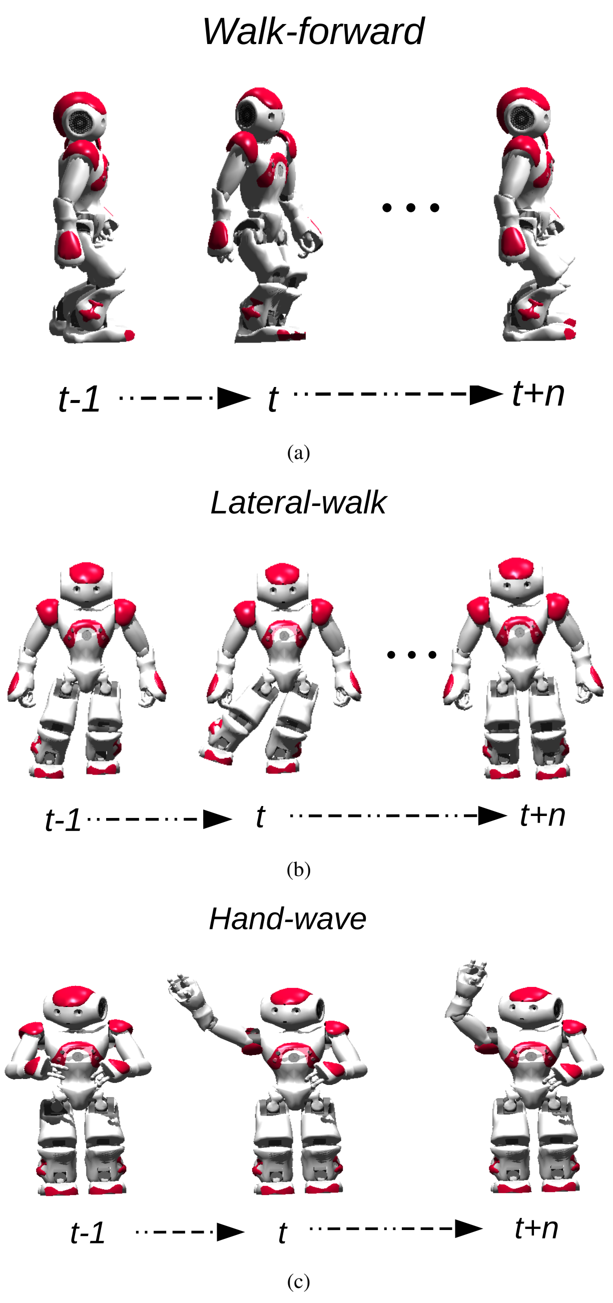


Fig. 5. Three types motions of a NAO robot. (a) shows a NAO robot walks forward, (b) is the lateral walk , (c) represents that a NAO robot waves right hand.

$(pc1, pc2, pc3)$ as shown in **Fig. 6**. $pc1$ represents the first component which has the largest variance in PCA and kPCA. $pc2$ and $pc3$ are the second and third components. **Figure 6(a)** is the motion model of walk-forward. Each point in the learned 3D motion model represents a pose state of the robot. An original robot pose is defined by 23 dimensional data. Through the learning, the robot pose is represented by 3D data. The trajectories in 3D latent space show how the robot motion changes along time t. The red arrows show the direction of the humanoid robot movements. Two latent points are marked out in **Fig. 6(a)** that one shows the robot pose at time t, consequently, the

Table 1. Cumulative contribution rate of PCA: using walk-forward source data

Dimension ($m-D$)	**cumulative contribution rate($\%$)**
$1D$	51.09
$2D$	83.12
$3D$	90.74
$4D$	96.68
$5D$	99.27
$6D$	99.64

next one is the pose state at time $t+1$. **Figure 6(b)** shows the motion model of lateral-walk. The hand-wave model is shown in **Fig. 6(c)**. Obviously, the models learned with standard-PCA-GPDM is not good as **Fig. 6** shows that the learned trajectories in 3D latent space are not smooth. Compared with **Figs. 6(a)** and **6(b)**, **Fig. 6(c)** is much better. However, as the black solid ellipse shows in **Fig. 6(c)**, the smoothness get worse. This also proves that the standard PCA method is not suitable for nonlinear motion data.

5.2. Auto-encoder-GPDM method

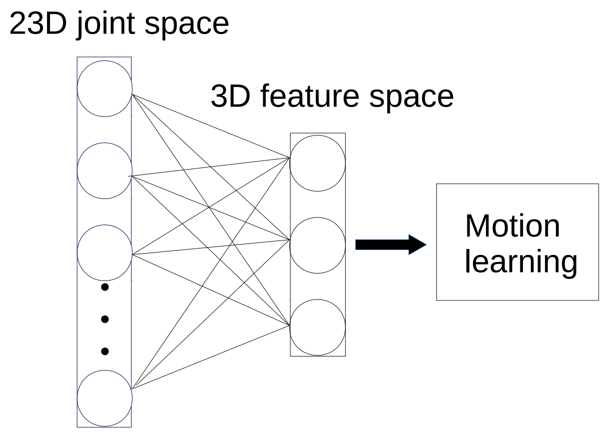


Fig. 7. Auto-encoder-GPDM

Yo Kondo and Yasutake Takahashi[9] used an auto-encoder method for address humanoid robot motion models. They only used an auto-encoder method rather than auto-encoder-GPDM method. In their research, they did not get a good result with auto-encoder addressing high dimensional humanoid robot motion data directly. Instead, the divided the humanoid robot body into four parts and constructed seven networks to realize dimensionality reduction. Taehwan Kim et al[8] used an auto-encoder-GPDM method for human motion learning. In our research, we also did experiments with the auto-encoder-GPDM method for high dimensional humanoid robot motion learning.

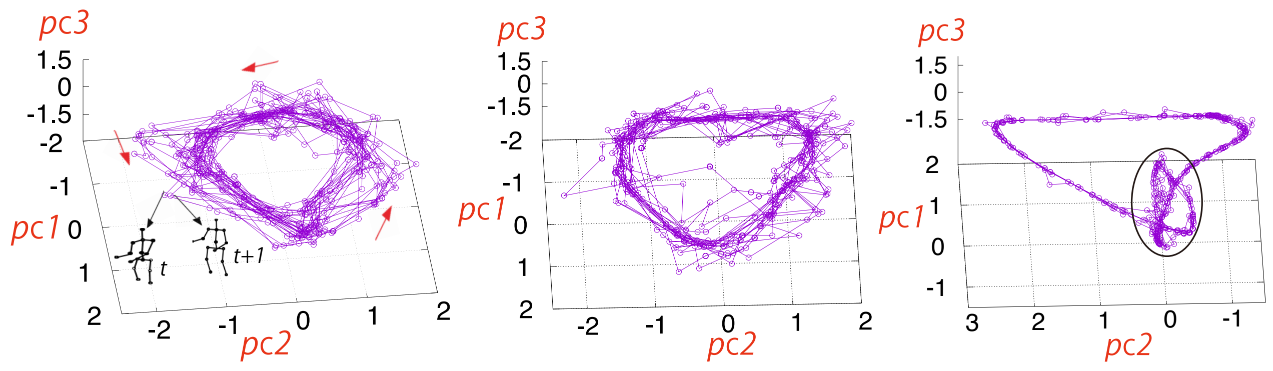
Figure 7 shows the auto-encoder-GPDM method we adopted in our experiments. In our experiments, we address the full humanoid robot motion data directly as **Fig. 7** shows. 23 dimensional humanoid robot motion data are mapped with auto-encoder directly as input data. Only one hidden layer with three neurons is used in our auto-encoder-GPDM. In this process, we extract the features into a low 3D feature space. Then, the features are learned and the motion models can be built. The motion models built with an auto-encoder-GPDM method are shown in **Fig. 8**. The 3D feature space is defined by $(fe1, fe2, fe3)$. $fe1$, $fe2$, $fe3$ are the features learned by auto-encoder-GPDM. It is easy to find that the motion models built with the auto-encoder-GPDM method are not smooth. This result is almost the same as Yo Kondo and Yasutake Takahashi[9] that they divided the full humanoid robot body into four parts instead of addressing the full robot body data.

5.3. Motion models built with kernel-PCA-GPDM method

Feature extraction is important in addressing high dimensional data set and machine learning. Good feature extraction methods help to learn good models in machine learning. In this paper, we investigate performances of kernel-PCA-GPDM method using different kernels. Through the analysis of the learned output motion models, we can figure out which kernel performs better in humanoid robot motion learning. Four different kernel functions defined from **Eq. (1)** to **Eq. (3)** are adopted in kernel PCA to extract high dimensional data feature of a NAO robot body joints into 3D low latent space. **Table 2** shows the cumulative contribution rate using kernel PCA with different kernels. In 2D latent space, all the cumulative contribution rates of the four types kernel functions are larger than 85% and less than 88%. To ensure that all the features are extracted into low latent space, we choose to extract high dimensional data into 3D latent space where the cumulative contribution rates are larger than 94$\%$. According to the cumulative contribution rate **Table 2**, it is easy to find out that the extracted features in 3D latent space can represent the original high dimensional motion data well with the cumulative contribution rate larger than 94$\%$.

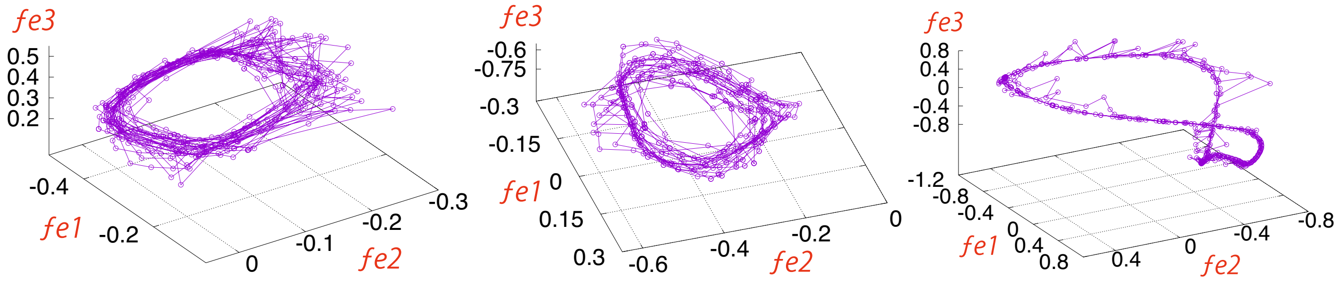
The humanoid robot motion models are learned with kernel-PCA-GPDM method in a low 3D latent space. **Figure 9** shows the motion models learned with a linear kernel-PCA-GPDM method. A very simple linear kernel function(**Eq. (1)**) is used. This linear kernel function does not need other parameters. Compared with **Fig. 6**, the motion models learned with linear kernel-PCA-GPDM method become smoother. However, it is still not smooth shown as **Fig. 9(a)** and **Fig. 9(b)**. Unfortunately, for the learned hand-wave model in **Fig. 9(c)**, the black solid ellipse part is almost nearly overlapped. It is difficult to express the motions of a robot with overlapped points because each point of the 3D space trajectory we learned stands for a pose state along time t. A good model should express the motions clearly and smoothly.

Figure 10 shows the three motion models learned with a polynomial kernel-PCA-GPDM method. The polyno-



(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 6. Motion models built with standard-PCA-GPDM method

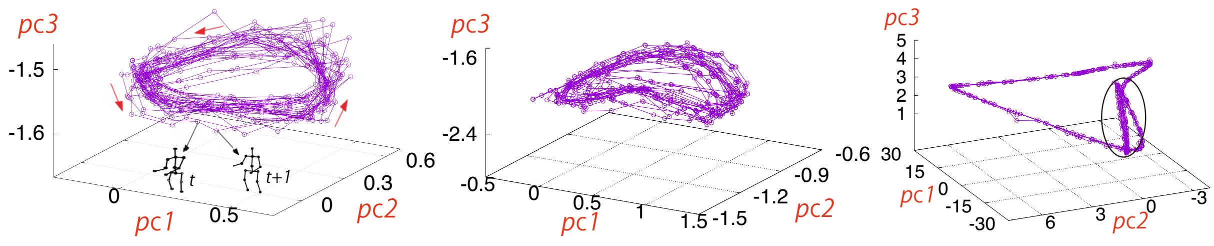


(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 8. Motion models built with auto-encoder-GPDM method

Table 2. Cumulative contribution rate of kernel PCA: using walk-forward source data.

Dimension	Kernel function			
$m-D$	Linear(%)	Polynomial(%)	RBF(%)	Hyperbolic Tangent(%)
1D	52.82	52.17	52.07	52.67
2D	87.83	85.78	86.95	87.35
3D	94.58	94.59	94.03	94.58
4D	97.77	97.69	97.31	97.75
5D	99.41	99.29	98.98	99.37
6D	99.63	99.57	99.21	99.61



(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 9. Motion models built with a linear kernel-PCA-GPDM method

mial kernel function is defined in **Eq. (2)**. There are three parameters ξ, c and e in polynomial kernel function. Here we use a common set $\xi = \frac{1}{M}$, $c = 0$, $e = 3$ where M is the dimension of input data set. e, we choose the dimension of latent space that the high dimensional motion data projected into. From **Fig. 10(a)** and **Fig. 10(b)**, we can get that the polynomial kernel-PCA-GPDM method did not work well on the walk-forward model and the lateral-walk model. It works well on the hand-wave model as shown in **Fig. 10(c)** even though some points are detached from the main trajectory where the points are circled in a solid black ellipse.

Figure 11 is the models learned with a Hyperbolic Tangent kernel-PCA-GPDM method. The Hyperbolic Tangent kernel is shown in **Eq. (3)**. In the learning process, $\xi = \frac{1}{M}$, $c = -1$. **Figure 11** shows that the Hyperbolic Tangent kernel-PCA-GPDM trains smoother motion models than the two kernels PCA GPDM we described before. Especially for the later-walk model and the hand-wave model, the Hyperbolic Tangent kernel-PCA-GPDM performs better. The lateral-walk motion model shown in **Fig. 11(b)** is much smoother compared to the three kernels PCA methods. For the hand-wave motion model, it has the same problem as the model learned with polynomial kernel-PCA-GPDM method where few points are detached from the main trajectory as shown in **Fig. 11(c)**. As for the walk-forward motion model, we aim to build a much smoother motion model.

In kernel functions, the RBF kernel is useful and is widely applied in machine learning. As we described in **Section 4.2**, the RBF kernel function is shown as **Eq. (4)**. The RBF kernel-PCA-GPDM method works well as **Fig. 12** shows. **Figure 12(a)** illustrates a smooth walk-forward motion model is learned. The lateral-walk motion model is also much smoother. **Figure 12(c)** shows a smooth hand-wave motion model. However, the trajectory shown in the ellipse are crowded and nearly overlapped. It is hard to express the hand-wave motion clearly with this motion model. In order to solve this problem, we put forward a novel kernel function combining an RBF and a Hyperbolic Tangent kernel.

5.4. A novel RBF-HT kernel-PCA-GPDM method

Among those four kernels PCA GPDM method, we found that the Hyperbolic Tangent kernel and the RBF kernel performs better than the other two kernels PCA GPDM methods. The method using RBF kernel build a much smoother motion models. However, as **Fig. 12(c)** shows, some latent points of the motion model are overlapped. In order to figure out a way to solve this problem, we develop a novel kernel function by combining an RBF kernel and a Hyperbolic tangent kernel where it is named RBF-HT kernel. The novel kernel function is defined in **Eq. (17)**. In the experiments, the parameters in **Eq. (17)** are set by empirical values where $\xi_1 = 0.1$, $\sigma = 4$, $\xi_2 = 20$, $\xi_3 = 0.04$, $c_1 = -1$, $c_2 = 0$. The result of the RBF-HT kernel-PCA-GPDM method is shown in **Fig. 13**. From the result, we can easily get that the RBF-HT kernel performs well. The ellipse part in **Fig. 12(c)** are overlapped. With the RBF-HT kernel method, the motion model can clearly illustrate the hand-wave motion as shown in **Fig. 13(c)**. This proves the efficiency of our proposed RBF-HT kernel method.

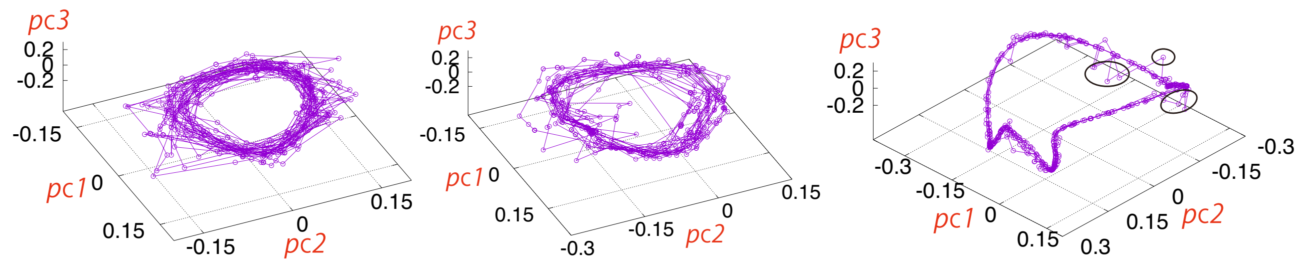
$$\begin{aligned} k(y,y') =& \xi_1 exp\left(-\frac{1}{2\sigma^2}\left\|y-y'\right\|^2\right)+ \\ & \xi_2 tanh(\xi_3 y^T y' + c_1) + c_2 \end{aligned} \quad . \ . \ . \ . \ (17)$$

6. Motion models built with RBF-HT kernel-PCA-BGPDM method

To make the motion models smoother, **Eq. (14)** is changed to a new equation

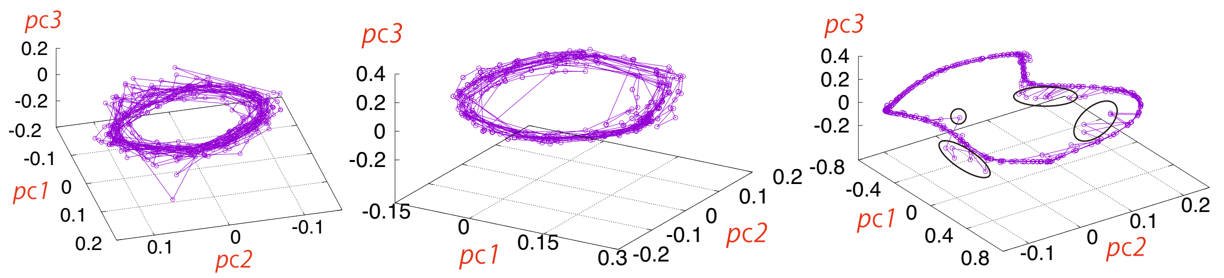
$$\begin{aligned} \mathscr{L} =& \frac{M}{2} ln|K_Y| + \frac{1}{2} tr(K_Y^{-1} Y W^2 Y^T) - N ln|W| + \\ & \frac{M}{d}\left(\frac{d}{2} ln|K_X| + \frac{1}{2} tr(K_X^{-1} X_o X_o^T)\right) \\ & + \sum_i ln\alpha_i + \sum_i ln\beta_i + \gamma. \end{aligned} \qquad (18)$$

The parameter $\frac{M}{d}$ is used for minifying the differences in the regressions to make the influence of dynamics and reconstruction could be balanced[14]. The RBF-HT kernel parameters configuration are same as **Fig. 13**. **Figure 14** shows the motion models learned with the RBF-HT kernel-PCA-BGPDM method. The trajectories in 3D latent space are much smoother than the method of GPDM especially for the two motion models, walk-forward model and lateral-walk model. The walk-forward motion model in **Fig. 14(a)** is a good motion model. As **Fig. 14(b)** shows, the lateral-walk motion model is also smooth except few points in the solid black ellipse. The hand-wave motion model in **Fig. 14(b)** is smoother than the model learned with GPDM. To learn a much smoother hand-wave motion model, we re-configure the parameters of the RBF-HT kernel where $\xi_1 = 0.1$, $\sigma = 3.46$, $\xi_2 = 0.1$, $\xi_3 = 0.04$, $c_1 = -0.1$, $c_2 = 0.01$. The result is shown in **Fig. 15**. **Figure 15(c)** illustrates that the new configuration of the RBF-HT kernel parameters works well. The hand-wave motion model is much smoother than the previous motion models. From **Figs. 15(a)** and **15(b)**, we can easily get that the new configuration of the RBF-HT kernel function performs well and generates smooth motion models. Even through the walk-forward and lateral-walk models are much smoother in **Fig. 14**, the motion models in **Fig. 15** are also smooth. Noises exist in lateral-walk motion model as shown in the solid black circle in **Fig. 15(b)**. We find that noises exist no matter how we re-configure the RBF-HT kernel parameters. Unlike a human, it is difficult for a NAO robot to keep its body balance and motion stability when it performs some motions. Usually, a NAO robot need to slow down the motion speed and narrow its movement range in order to keep its body balance and motion stability. Even though, there's more or less jerking when a robot performs human-like motions. It cannot be avoided that



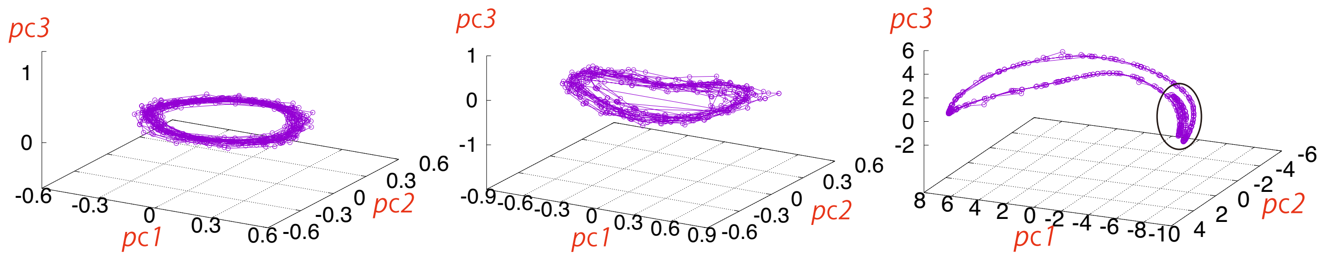
(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 10. Motion models built with a polynomial kernel-PCA-GPDM method



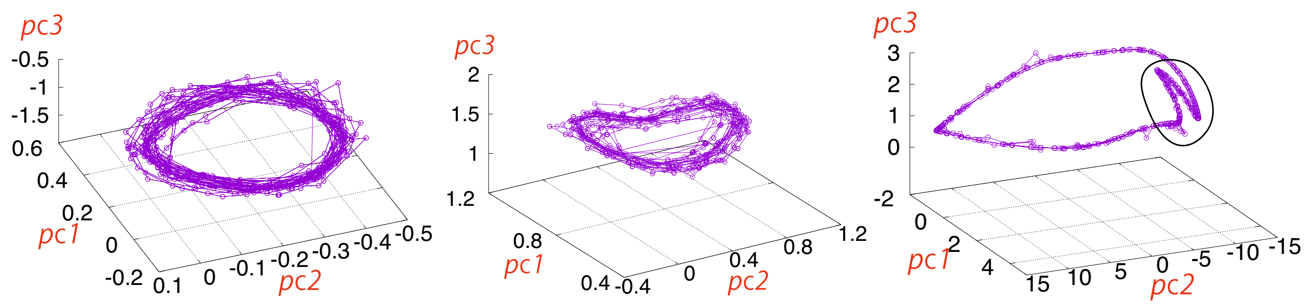
(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 11. Motion models built with a Hyperbolic Tangent kernel-PCA-GPDM method



(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 12. Motion models built with an RBF kernel-PCA-GPDM method



(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 13. Motion models built with the RBF-HT kernel-PCA-GPDM method

noises exist when a NAO robot lose its balance in collecting motion data. Compared with walk-forward and hand-wave motions, the lateral-walk motion is more difficult for the NAO robot to maintain its balance and motion stability while perform the motion precisely. Noises still exist after learning with kernel-PCA-GPDM. The NAO robot keeps better body balance and motion stability in walk-forward and hand-wave motions so that we hardly find noises in walk-forward motion model(**Fig. 14(a)**) and hand-wave motion model(**Fig. 15(c)**). The result is consistent with the humanoid robot motion data. Above all, the motion models learned with the RBF-HT kernel-PCA-GPDM method work efficiently in humanoid robot motion model learning.

7. Humanoid robot motion representation

To verify the motion models learned with kernel-PCA-GPDM method, we restore the learned 3D motion data back to 23 dimension observation space. In this way, we realize humanoid robot motion representation. The restored joints data are applied into a NAO robot in simulation. Here we choose walk-forward model as an example. Through the simulation, we find that the NAO robot walks more stably with the learned motion data. The robot walks with less effect of noise data. This proves the efficiency of our learned motion models. We also compare the restored motion joints data with the original motion joints data. **Figure 16** shows some important joints angles. The purple line shows the original training joints angles and the green line show the restored joints angles using kernel-PCA-GPDM method. From **Fig. 16(a)** and **Fig. 16(b)**, it is obviously that the restored data is almost overlapped with the original data for robot joints LShoulderPitch and RShoulderPitch. For both left and right KneePitch joints and AnklePitch joints, the restored data are also overlapped with the original data. The restored joints angles become smoother than the original data. This is why the robot moves more stably. As **Fig. 16(c), Fig. 16(d), Fig. 16(e)** and **Fig. 16(f)** show that the range of restored joints angles is smaller than the original data for some joints even though the motion model we build is efficient. This is the reason that the movement range of some robot joints becomes smaller compared with the original joints movement range.

8. Conclusions and Future work

In this paper, we use kernel-PCA-GPDM method for training full body robots joints data and building humanoid robot motion models with a small amount of motion data. A novel RBF-HT kernel is proposed to build smooth motion models in kernel-PCA-GPDM. To achieve good motion models, we first reduce the 23 high dimensional motion data into 3D latent space. Then, through the learning of kernel-PCA-GPDM method, we build efficient humanoid motion models with small numbers of motion data. In the experiments, different kernel-PCA-GPDM methods are invested to find out which method performs better. Besides, we compare our kernel-PCA-GPDM method with standard-PCA-GPDM method. Further more, we execute a comparison between our kernel-PCA-GPDM method and an auto-encoder-GPDM method. The comparison indicates that our proposed kernel-PCA-GPDM method builds humanoid robot motion models efficiently. A kernel PCA balanced GPDM method is adopted to optimize the motion models. Smooth motion models are learned with the RBF-HT kernel-PCA-BGPDM method. Finally, the learned 3D motion features are restored back to 23 dimension space. The results verified that our method could build humanoid robot motion models efficiently. In the future, we will apply the learned 3D motion models into real-time imitation system in order realize quick exploration in low latent space. Besides, we want to build much more humanoid robot motion models through an imitation system, especially for some complex motions. Based on the humanoid robot motion models, we will make the robot recognize and classify human motions in imitation systems. As our motion models are dynamically learned, the robot motion models can be used to predict next state of human motions after the robot recognize human motion.

Acknowledgements
The authors would like to thank those who has made great assistance to our study.

References:

[1] Y. Ou, J. Hu, Z. Wang, Y. Fu, X. Wu, and X. Li, "A Real-Time Human Imitation System Using Kinect", International Journal of Social Robotics, 7(5):587–600, 2015.

[2] S. Morante, J. G. Victores, A. Jardón, and C. Balaguer, "Humanoid robot imitation through continuous goal-directed actions: an evolutionary approach", Advanced Robotics, 29(5):303–314, 2015.

[3] J. Lei, M. Song, Z.-N. Li, and C. Chen, "Whole-body humanoid robot imitation with pose similarity evaluation", Signal Processing, 108:136 – 146, 2015.

[4] W. He, W. Ge, Y. Li, Y. J. Liu, C. Yang, and C. Sun, "Model Identification and Control Design for a Humanoid Robot", IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(1):45–57, Jan 2017.

[5] P. Shahverdi and M. T. Masouleh, "A simple and fast geometric kinematic solution for imitation of human arms by a NAO humanoid robot", In 2016 4th International Conference on Robotics and Mechatronics (ICROM), pp. 572–577, Oct 2016.

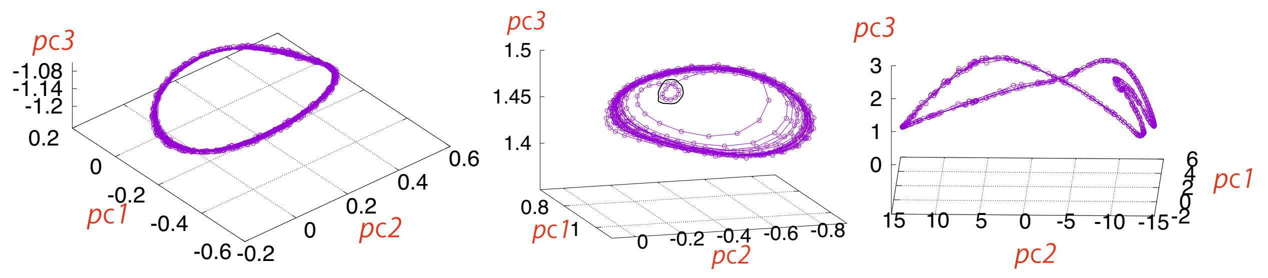
[6] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian Process Dynamical Models for Human Motion", IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2):283–298, Feb 2008.

[7] N. Yamaguchi, "Visualizing State of Time-Series Data by Supervised Gaussian Process Dynamical Models", Journal of Advanced Computational Intelligence and Intelligent Informatics, 21(5):825 – 831, 2017.

[8] T. Kim, J. Park, S. Heo, K. Sung, and J. Park, "Characterizing Dynamic Walking Patterns and Detecting Falls with Wearable Sensors Using Gaussian Process Methods", Sensors, 17(5), 2017.

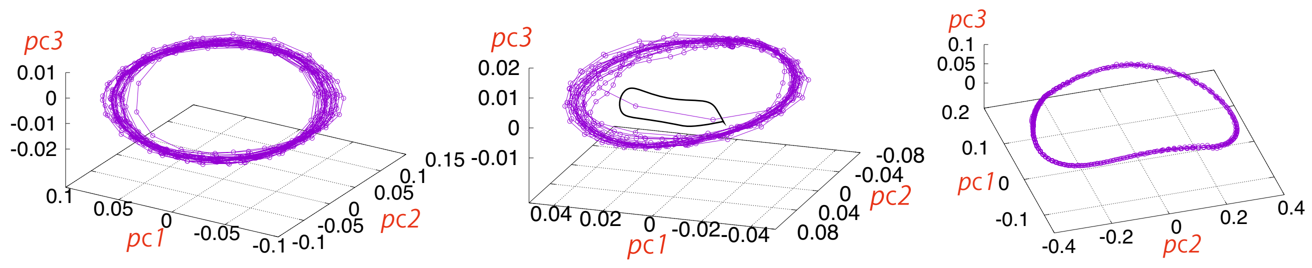
[9] Y. Kondo, S. Yamamoto, and Y. Takahashi, "Real-Time Posture Imitation of Biped Humanoid Robot Based on Particle Filter with Simple Joint Control for Standing Stabilization", In 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS), pp. 130–135, Aug 2016.

[10] K. S. Hwang, W. C. Jiang, and Q. H. Yang, "Posture imitation and balance learning for humanoid robots", In 2016 International Conference on Advanced Robotics and Intelligent Systems (ARIS), pp. 1–6, Aug 2016.



(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 14. Motion models built with the RBF-HT kernel-PCA-BGPDM method using re-configuration parameters



(a) Walk-forward (b) Lateral-walk (c) Hand-wave

Fig. 15. Motion models built with the RBF-HT kernel-PCA-BGPDM method

[11] Y. Takahashi, Y. M. Hiroki Hatano, and Y. M. Kazuyuki Usui, “Motion Segmentation and Recognition for Imitation Learning and Influence of Bias for Learning Walking Motion of Humanoid Robot Based on Human Demonstrated Motion”, Journal of Advanced Computational Intelligence and Intelligent Informatics, 19(4), 2015.

[12] H. Kang and F. C. Park, “Humanoid motion optimization via nonlinear dimension reduction”, In 2012 IEEE International Conference on Robotics and Automation, pp. 1444–1449, 2012.

[13] L. Raskin, M. Rudzsky, and E. Rivlin, “Dimensionality reduction using a Gaussian Process Annealed Particle Filter for tracking and classification of articulated body motions”, Computer Vision and Image Understanding, 115(4):503 – 519, 2011.

[14] R. Urtasun, D. J. Fleet, and P. Fua, “3D People Tracking with Gaussian Process Dynamical Models”, In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), volume 1, pp. 238–245, June 2006.

[15] S. Sedai, M. Bennamoun, D. Q. Huynh, A. El-Sallam, S. Foo, J. Alderson, and C. Lind, “3D human pose tracking using Gaussian process regression and particle filter applied to gait analysis of Parkinson’s disease patients”, In 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), pp. 1636–1642, June 2013.

[16] N. D. Lawrence, “Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data”, In S. Thrun, L. K. Saul, and B. Schölkopf, editors, Advances in Neural Information Processing Systems 16, pp. 329–336. MIT Press, 2004.

[17] J. Mi and Y. Takahashi, “Motion representation for humanoid robot based on Gaussian Process Dynamical models”, In The 5th International Workshops on Advanced Computational Intelligence and Intelligent Informatics(IWACIII2017), Nov.2-5, 2017, 2017.

[18] B. Schölkopf and A. J.Smola, ”Learning with Kernels”, The MIT Press, Cambridge, Massachusetts, London, England, 2001.

[19] I. Rodríguez, A. Aguado, O. Parra, E. Lazkano, and B. Sierra, ”NAO Robot as Rehabilitation Assistant in a Kinect Controlled System”, pp. 419–423, Springer International Publishing, Cham, 2017.

[20] C. Li and X. Wang, “Visual localization and object tracking for the NAO robot in dynamic environment”, In 2016 IEEE International Conference on Information and Automation (ICIA), pp. 1044–1049, Aug 2016.

[21] Z. B. Wang, L. Yang, Z. P. Huang, J. K. Wu, Z. Q. Zhang, and L. X. Sun, “Human motion tracking based on complementary Kalman filter”, In 2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN), pp. 55–58, May 2017.

[22] Y. Deng, T. Mao, M. Shi, and Z. Wang, “Cloth Deformation Prediction Based on Human Motion”, In 2016 International Conference on Virtual Reality and Visualization (ICVRV), pp. 258–263, Sept 2016.

[23] B. Jokanovic, M. Amin, and B. Erol, “Multiple joint-variable domains recognition of human motion”, In 2017 IEEE Radar Conference (RadarConf), pp. 0948–0952, May 2017.

[24] H. Zhu, Y. Yu, Y. Zhou, and S. Du, “Dynamic Human Body Modeling Using a Single RGB Camera”, Sensors, 16, 2016.

[25] J. Bütepage, M. J. Black, D. Kragic, and H. Kjellström, “Deep representation learning for human motion prediction and classification”, CoRR, abs/1702.07486, 2017.

[26] F. Hu, W. Liu, X. Wu, and D. Luo, “Learning basic unit movements with gate-model auto-encoder for humanoid arm motion control”, In 2016 IEEE International Conference on Information and Automation (ICIA), pp. 246–251, Aug 2016.

[27] P. Xu, M. Ye, Q. Liu, X. Li, L. Pei, and J. Ding, “Motion detection via a couple of auto-encoder networks”, In 2014 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6, July 2014.

[28] R. Urtasun, D. J. Fleet, and P. Fua, “3D People Tracking with Gaussian Process Dynamical Models”, In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), volume 1, pp. 238–245, June 2006.

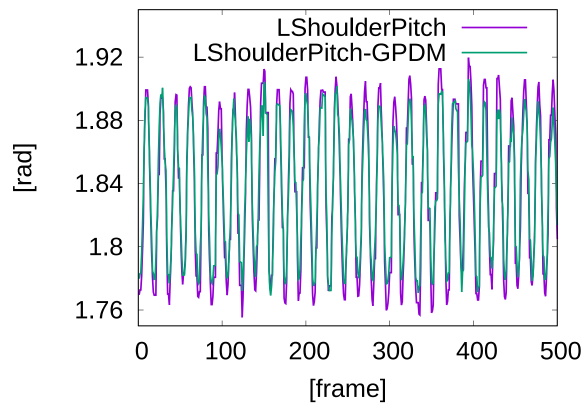
[29] N. Yamaguchi, “Visualizing States of Time-Series Data by Autoregressive Gaussian Process Dynamical Models”, Journal of Advanced Computational Intelligence and Intelligent Informatics, 21(5), 2017.

[30] I. T. Jolliffe, ”Principal Component Analysis”, 0172-7397, Springer New York, 2002.

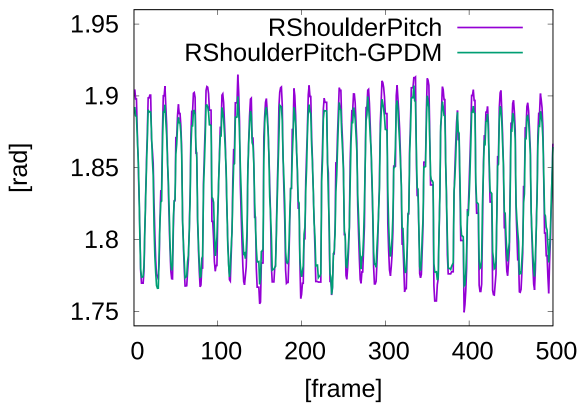
[31] J. Su, D. Yi, C. Liu, L. Guo, and W.-H. Chen, “Dimension Reduction Aided Hyperspectral Image Classification with a Small-sized Training Dataset: Experimental Comparisons”, Sensors, 17(12), 2017.

[32] R. M. Neal, ”Bayesian Learning for Neural Networks”, 0930-0325, Springer New York, 1996.

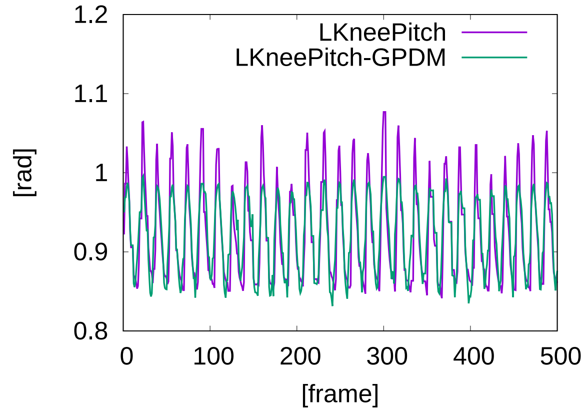
[33] D. J. Mackay, ”Information Theory, Inference, and Learning Algorithms”, Cambridge University Press, 2003.



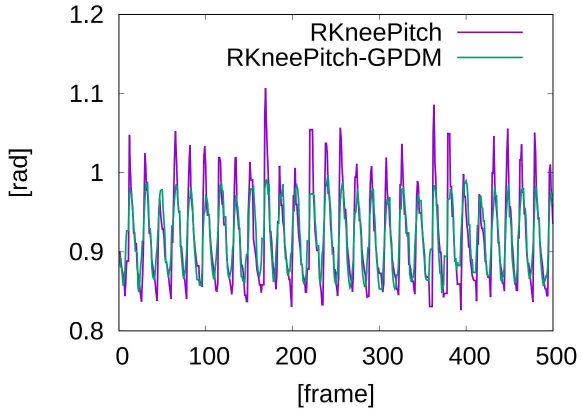
(a) LShoulderPitch



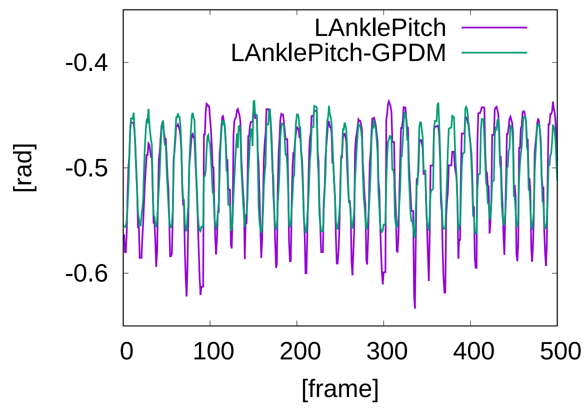
(b) RShoulderPitch



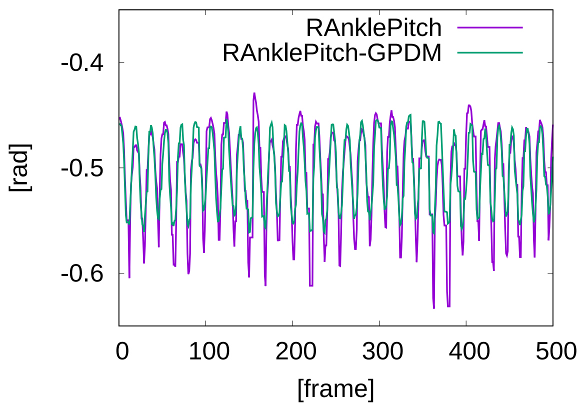
(c) LKneePitch



(d) RKneePitch

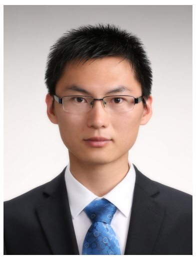


(e) LAnklePitch



(f) RAnklePitch

Fig. 16. Robot joints angle representation: the purple line is the training joints angle and the green line is the representation



Name:
Jian Mi

Affiliation:
Department of Human and Artificial Intelligent Systems, Graduate School of Engineering, University of Fukui

Address:
3-9-1, Bunkyo, Fukui, Fukui, 910-8507, Japan

Brief Biographical History:
2009-2013 Beijing Information Science and Technology University
2014- Graduate School of Engineering, University of Fukui

Main Works:
•J. Mi and Y. Takahashi, Design of an HF-Band RFID System with Multiple Readers and Passive Tags for Indoor Mobile Robot Self-Localization, Sensors, Vol.16, No.8, p.1200, 2016
J. Mi and Y. Takahashi, Low Cost Design of HF-band RFID System for Mobile Robot Self-Localization based on Multiple Readers and Tags, Proceedings of 2015 IEEE International Conference on Robotics and Biomimetics(ROBIO2015), pp.194-199, DOI: 10.1109/ROBIO.2015.7418766
Jian Mi and Yasutake Takahashi, Performance Analysis of Mobile Robot Self-Localization Based on Different Configurations of RFID System, Proceedings of 2015 IEEE/ASME International Conference on Advanced Intelligent Mechatronics(AIM2015), pp.1591-1596, DOI: 10.1109/AIM.2015.7222770

Name:
Yasutake Takahashi

Affiliation:
Department of Human and Artificial Intelligent Systems, Graduate School of Engineering, University of Fukui

Address:
3-9-1, Bunkyo, Fukui, Fukui, 910-8507, Japan

Brief Biographical History:
2000-2009 Assistant Professor of Department of Adaptive Machine Systems, Graduate School of Engineering, Osaka University
2006-2007 Visiting researcher at the Fraunhofer IAIS
2009-2012 Senior Assitant Professor of Department of Human and Artificial Intelligent Systems, Graduate School of Engineering, University of Fukui
2012- Associate Professor of Department of Human and Artificial Intelligent Systems, Graduate School of University of Fukui

Main Works:
• Y. Takahashi, K. Noma, and M. Asada. Efficient Behavior Learning based on State Value Estimation of Self and Others. Advanced Robotics, Vol.22, No.12, pp.1379-1395, 2008
Y. Takahashi, Y. Tamura, M. Asada, and M. Negrello, “ Emulation and Action Understanding through Shared Values ”, ROBOTICS AND AUTONOMOUS SYSTEMS Journal, Vo.58, No.7, pp.855-865, 2010
Y. Tamura, Y. Takahashi, M. Asada, “ Observed Body Clustering for Imitation Based on Value System ”, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.14, No.7, pp.802-812, 2010

Membership in Academic Societies:
• The Robotics Society of Japan (RSJ)
Japan Society for Fuzzy Theory and Intelligent Informatics (SOFT)
The Japanese Society for Artificial Intelligence (JSAI)
The Institute of Electrical and Electronics Engineers (IEEE)