OSAKA UNIVERSITY

Learning Periodic Human Motion Through Imitation Using Eigenposes

by

Rawichote Chalodhorn

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the

Graduate School of Engineering
Department of Adaptive Machine Systems

March 2009

Thesis Supervisor: Prof. Minoru Asada

Thesis Committee: Prof. Minoru Asada, Chair
Prof. Hiroshi Ishiguro
Assoc. Prof. Koh Hosoda

OSAKA UNIVERSITY

Abstract

Graduate School of Engineering
Department of Adaptive Machine Systems

Doctor of Philosophy

by Rawichote Chalodhorn

This dissertation provides the first demonstration that a humanoid robot can learn to perform human dynamic motion such as walking directly by imitating a human gait obtained from motion capture data without any prior information of its dynamics model. Programming a humanoid robot to perform an action that takes into account the robot's complex dynamics is a challenging problem. Traditional approaches typically require highly accurate prior knowledge of the robot's dynamics and environment in order to devise complex (and often brittle) control algorithms for generating a stable dynamic motion. Training using human motion capture is an intuitive and flexible approach to programming a robot but direct usage of mocap data usually results in dynamically unstable motion. Furthermore, optimization using mocap data in the humanoid full-body joint-space is typically intractable. This dissertation purposes a new model-free approach to tractable imitation-based learning in humanoids. Kinematic information from human motion capture is represented in a low dimensional subspace. Motor commands in this low-dimensional space are mapped to sensory feedback to learn a predictive dynamic model. This model is used within an optimization framework to estimate optimal motor commands that satisfy the initial kinematic constraints as best as possible while at the same time generating dynamically stable motion. The viability of this approach is demonstrated by providing examples of dynamically stable walking learned from motion capture data using both a simulator and a real humanoid robot.

Acknowledgements

Firstly at Asada Lab, I would like to thank my thesis advisor Prof. Minoru Asada for the initial direction of this research. It was also his visionary that the eigenpose data can be used for a humanoid robot to learn human motion. Without him, there is no this discovery. Thank Dr. Karl F. MacDorman for working on the preliminary study of NLPCA and CNLPCA together. Thank Dr. Koh Hosoda for intensive technical and theoretical discussion. Thank Dr. Yasutake Takahashi for being a mentor for Linux programming and RoboCup participation. The RoboCup humanoid league was also where the research in this dissertation took off, so thank RoboCup. Thank Dr. Masaki Ogino for being my mentor on HOAP robots programming. Thank every Ph.D. candidate at Asada Lab of year 2000-2003 for a lot of useful discussion. Thank Okada-san who helped me about many japanese document works in Asada Lab. And thank other Asada Lab's members as well.

Secondly at the department of computer science & engineering, University of Washington, thank Prof. Rajesh Rao for the opportunity to work at CSE UW, a lot of moral supports and great ideas of the state predictor in this work. Thank David Grimes and Gabriel Maganis for deep discussion of this research for every Monday to Friday for three years. Thank Keith Crochow for contribution of his expertise of motion capture system. And for moral support from people at CSE, UW.

Thirdly my family, my dad and my mom sacrificed many things in their life for me to get the best education possible. The words "thank you" will never be enough for their true care, true love, true support and true encouragement. And thank my younger brother Golf for taking good care of my mom and my dad all the time was not there.

Fourthly Jim Young & Carol Young, thank for your foods and love from your family in Seattle. My great ideas always come when my heart is warm and my stomach is full.

Last be not least to readers, thank you for reading 8-years of my work. There are a lot of rooms to improve this work. The ideas in this work are very novel but palusible. I strongly encourage you to extend this work and feel free to contact me.

Contents

List of Figures

List of Tables

Abbreviations

1-D	one-dimensional
3-D	three-dimensional
CNLPCA	circular constraint nonlinear principal components analysis
COM	center of mass
DOF	degrees of freedom
IK	inverse kinemaic
mocap	motion capture
NARX	nonlinear autoregressive network with exogenous inputs
NLPCA	nonlinear principal components analysis
PCA	principal components analysis
RL	reinforcement learning
ZMP	zero-moment point

Chapter 1

Introduction

Imitation is an important learning mechanism in many biological systems including humans [1]. Learning through imitation is a powerful and versatile method for acquiring new behaviors. In humans, a wide range of behaviors, from styles of social interaction to tool use, are passed from one generation to another through imitative learning. Unlike trial-and-error-based learning methods such as reinforcement learning (RL) [2], imitation is a allows fast learning. Learning by imitation is learning from demonstrations. Solutions of the learning problem were already shown to the learning agent. Thus for imitation learning, the learning agent only has to search for the optimal solution within a small search space. The potential for rapid behavior acquisition through demonstration has made imitation learning an increasingly attractive alternative to manually programming robots. It is straightforward to recover kinematic information from human motion using, for example, motion capture, but imitating the motion with stable robot dynamics is a much harder problem. Because, not only the kinematic problem is inherited in imitating dynamically stable motion, but imitation of dynamically stable motion also involved with deriving appropriate action commands based on dynamic interaction between the robot and its environment. Sensory feedback data also must be taken into account for solving the dynamic problem.

Traditional model-based approaches based on zero-moment point (ZMP) [3–5] or the inverted pendulum model [6, 7] require a highly accurate model of robot dynamics and the environment in order to achieve a stable walking gait. Learning-based approaches such as RL are more flexible and can adapt to environmental change but such methods are typically not directly applicable to humanoid robots due to the curse of dimensionality problem engendered by the high dimensionality of the full-body joint space of the robot. Morimoto et al. [8] demonstrated that stepping and walking policies could be improved by using RL method on the extracted feature space by using kernel dimension

reduction (KDR). Their result is shown in a dynamic simulator. This work is a fruitful result of studying nonlinear dynamics of passive dynamic walking mechanisms [9–11]. In their work, the nominal stepping and walking controller are provided, and their learning system improves the performances of these controllers. Our work uses less assumptions. An assumption of a specific type of nonlinear dynamical system is not employed in the framework in this dissertation like in [8]. Only Markovian causal relationship of state and action is assumed. The motion imitation framework in this dissertation is designed for learning general human motion from demonstrations. It can be used for learning different gaits for different tasks without redesign the algorithm.

The approach in this dissertation builds on several previous approaches to humanoid motion generation and imitation. Tatani and Nakamura [12] first applied non-linear principal components analysis (NLPCA) [13] to human and humanoid robot motion data. The work shows that the motions can be kinematically reproduced from low-dimensional data. The Gaussian Process Dynamical Models (GPDM) by Wang et al. [14], is a dimensionality reduction method for modeling high-dimensional sequential data. GPDM can be analogized to Gaussian process latent variable models (GPLVM) when temporal sequence of data is taken into account. In this work, a temporal sequence of human walking data motion was modeled and reproduced without prior information. Dynamic modeling in GPDM contexts is modeling of temporal sequence of a data pattern. The word dynamic in contexts of GPDM does not have the same meaning as dynamic in robotics, which involving with properties that cause interaction between the robot and its environment such as force, torque, mass and moment of inertia. The word dynamic in contexts of GPDM is dynamic in computer science aspect, which refers to sequential data, as opposed to the word static in computer science aspect which refers to concurrent data. The resulted walking gait of GPDM in this work was reproduced kinematically in robotics aspect. Low-dimensional data of walking postures in GPDM latent space do not interact with environment. Our motion learning framework in this paper, learns a dynamic model of interaction between the robot and its environment through causal relationship of sensory feedback and low-dimensional posture commands.

The idea of using imitation to train robots also has been explored by a number of researchers. In 1994 Demiris and Hayes [15] introduced the concept of imitative learning by demonstrating a wheeled mobile robot that learned to solve a maze problem by imitating another homologous robot. In 1999 Billard [16] showed that imitation is a mechanism that allows the robot imitator to share a similar set of proprio- and exteroceptions with teacher. Ijspeert et al. [17] designed a nonlinear dynamical system to imitate trajectories of joints and end-effectors of a human teacher. In this work, the robot learned and performed tennis swing motions by imitation. The resulted motions shows robust arm motion against dynamic perturbation. The mimesis theory of [18] is based on action

acquisition and action symbol generation but does not address dynamics compensation for real-time biped locomotion.

1.1 Human motion imitation framework

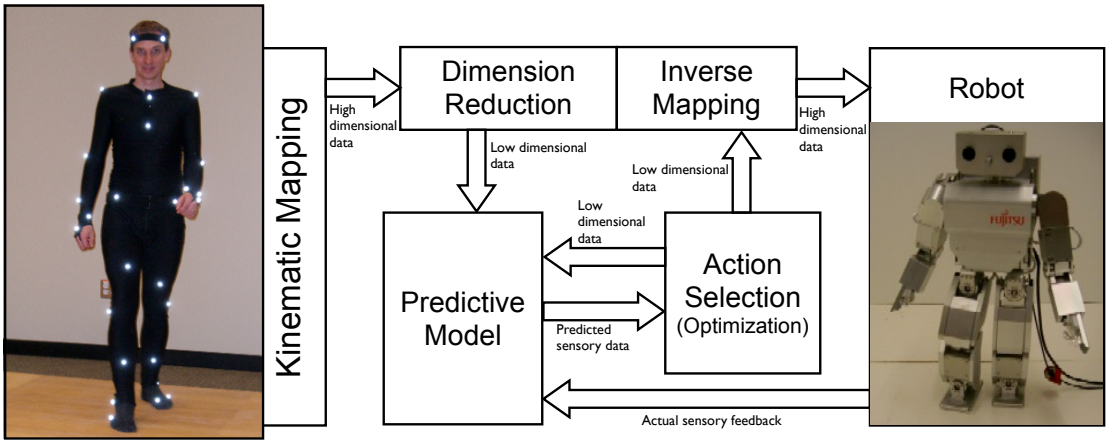

FIGURE 1.1: A framework for learning human behavior by imitation through sensory-motor mapping in reduced dimensional spaces.

In this dissertation, an approach to achieving stable gait acquisition in humanoid robots via imitation is proposes. The framework of the proposing method is shown in Figure 1.1. First, a motion capture system transforms Cartesian position of markers attached to the human body to joint angles based on kinematic relationships between the human and robot bodies. Then, linear PCA as dimensionality reduction to represent posture information in a compact low-dimensional subspace is employed. Optimization of whole-body robot dynamics to match human motion is performed in the low dimensional subspaces. In particular, sensory feedback data are recorded from the robot during motion and a causal relationship between actions in the low dimensional feature space and the expected sensory feedback is learned. This learned sensory-motor mapping allows humanoid motion dynamics to be optimized. An inverse mapping from the reduced space back to the original joint space is then used to generate optimized motion on the robot. Several results demonstrating that the proposed approach allows a humanoid robot to learn to walk based solely on human motion capture without the need for a detailed physical model of the robot are presented in this dissertation.

1.2 Dissertation outlines

Chapter 2 is a preliminary study of low-dimensional humanoid motion data, which is the motivation for the human motion learning framework in this dissertation. The sequence from a 25 degree-of-freedom humanoid robot performing a ball tracking task is reduced to its intrinsic dimensionality by nonlinear principal component analysis. The study demonstrates how a sequence of low-dimensional motion data can be automatically segmented into a set of circular patterns. Each circular pattern is correspondent with basic behavior in the motion sequence. Within the study, all of the segmented motion can be reproduced kinematically. A question was arisen at this point that is it possible to reproduce a dynamically stable motion from these segmented patterns. If each of basic motion such as walking straight, turning and sidestepping can be reproduced dynamically, it could be used as a high-level action commands. These action commands can be cooperated with visual information to learn a complex behavior such as ball following by a learning algorithm such as reinforcement learning. Though, a motion sequence of a complex behavior was successfully automatically segmented in the low-dimensional subspaces by using NLPCA and circular constrained NLPCA (CNLPCA) [19] cooperatively, pros and cons of algorithms that were implemented in the studying were found. The advantages were carried on and the disadvantages were improved in order to develop a methodology for learning dynamically stable human motion.

One significant disadvantage of NLPCA algorithm for dimensionality reduction is that construction of the low-dimesional subspaces is a very time consuming process. A fast and straightforward algorithm was chosen over NLPCA. In Chapter 3, the low-dimensional subspaces were created by linear PCA. Properties of human motion in low-dimensional subspaces, which is obtained from PCA will be described. Details of procedures such as data preprocessing for PCA, PCA transformation, inverse PCA transformation and basic definitions of this framework such as action subspace embedding and action subspace scaling are described in this chapter. The action subspace embedding is a set of posture commands that embed in the low-dimensional spaces. By using action subspace embedding, a complete motion cycle of a periodic motion can be generated from a single parameter function by varying the parameter from 0 to 2π. This function is also used for constructing the search-space for motion optimization in later chapters. The action subspace embedding was designed imitates a characteristic of CNLPCA for having a single angular parameter that can reproduce a motion pattern. However, an inability of CNLPCA when using for modeling a highly irregular closed-curve pattern was improved in action subspace embedding. The action subspace scaling is scaling the size action subspace embedding in the low-dimensional subspaces, which creates a

similar motion with different scale of magnitude of movement. This property is used for creating a smaller scale movement of a motion, which is more dynamically stable.

The motion learning process is explained in Chapter 4. The motion learning strategy uses a model-predictive motion generator plans an optimal complete cycle of motion based on sensory feedback prediction of a predictive model. The predictive model is constructed through a learning algorithm. The time-delay RBF network [20], which is a learning technique of time-series prediction is implemented for the predictive model. The predictor derives a state-value of sensory feedback from robot movement of one time-step ahead in future based on history information of motion commands from the low-dimensional subspaces and sensory feedback. Then, a motion optimization algorithm searches for an optimal low-dimensional action command, which is subjected to an objective function to build a cycle of motion. Examples of one-dimensional optimization and three-dimensional optimization of a hand-coded walking gait are shown in this chapter.

Learning of human motion through imitation is introduced in Chapter 5. A method of kinematic mapping of human motion data to a robot body is introduced to achieve learning human motion through imitation. A straight-forward human walking motion is the target motion to imitate. The action subspace scaling technique that described in Chapter 3 is employed to obtain a stable walking gait to start the learning process. Results of learning the human walking gait by using the three-dimensional motion optimization is demonstrated in this chapter.

In Chapter 6, an extension for optimization of the motion data in the low-dimensional subspaces beyond 3-D is introduced. The sidestep human motion is used as the target imitated motion. In this chapter, sidestep motion data from PCA transformation without dimensionality reduction are optimized to demonstrate an extreme example that the 100% accuracy of original posture of the motion can be recovered and complied with the motion learning framework. All of the results and the overall framework are discussed and concluded together in Chapter 7.

Chapter 2

Preliminary Study of Low-dimensional Humanoid Motion Data

The aim of developing a humanoid robot is to have a robot that can work cooperatively with people. Recently, robotics researchers have succeeded in developing mechanical platforms for humanoid robots. These robots can walk and perform simple tasks. However, these demonstrations are usually directed by conventional computer programs that are prepared under specific environmental conditions. The robot may not be able to perform properly, if the conditions change. Moreover, a humanoid robot must take account of too many conditions to perform versatile tasks, and a programmer cannot anticipate and prepare for all of these conditions [21]. One solution is to develop a robot that can learn to perform in a human environment.

Reinforcement learning provides a useful method of adapting to environmental change based on experience. The self-organizing, modular and hierarchical structure of multi-layered reinforcement learning extends reinforcement learning to more complicated problems [22]. There are drawbacks to applying conventional reinforcement learning to a real robot: the requirement of a long learning period and a well-designed state-action space. By introducing a set of examples to a reinforcement learning system, the learning time can be shortened [23]. A heuristic algorithm is applied to a sample set to generate a state-action space and learning modules automatically. The learning modules are also reusable for learning new complex behavior [24]. The reinforcement learning method works well with simple robots such as wheeled robots. However, for a humanoid robot that has a large number of actuators, existing reinforcement learning schemes cannot deal with its huge state-action space directly. One solution is to apply an abstract

state-action space to the hierarchical multi-module reinforcement learning method [24] instead of using the raw state-action space determined by the sensors and effectors.

An approach that segments humanoid motion data automatically is studied in this chapter. The segmentation results can be used as abstract states and abstract actions to facilitate the learning of complex tasks by hierarchical multi-module reinforcement learning method [24]. Nonlinear principal component analysis was used for reducing the high-dimensional space of humanoid motion data to a tractable three-dimensional feature space. Then, the algorithm incrementally employs CNLPCA to learn the data points and divide them into segments. A CNLPCA neural network tries to learn as many data point in temporal order as its learning capacity can accept. Once the learning capacity of a network is saturated, the network defines a segment and a new CNLPCA neural network is employed. The algorithm keeps applying CNLPCA neural networks to the data in temporal order until the end of the data is reached. As a result, different data patterns are automatically divided into segments, which match the original patterns.

Some redundant segments may occur in the segmentation result. The algorithm also minimizes the number of redundant segments by merging segments that are very close to each other based on the distance between the segments. As a result, automatically segmented trajectories characterize all the periodic motion patterns.

2.1 Nonlinear principal component analysis with a circular constraint

The human body has 244 degrees of freedom [25] and a vast array of proprioceptors. Excluding the hands, humanoid robots generally have at least 20 degrees of freedom. They are considered high-dimensional systems to which conventional learning algorithm cannot be applied. Fortunately, from the standpoint of a particular activity, the effective dimensionality may be much lower.

Given a coding function $f : \mathbb{R}^N \mapsto \mathbb{R}^P$ and decoding function $g : \mathbb{R}^P \mapsto \mathbb{R}^N$ that belong to the sets of continuous nonlinear function $\mathcal{C}$ and $\mathcal{D}$, respectively, where $P < N$ nonlinear principal component networks minimize the error function E :

$$\|\vec{x} - g(f(\vec{x}))\|^2, \quad \vec{x} \in \mathbb{R}^N \tag{2.1}$$

resulting in P principal components $[y_1 \cdots y_p] = f(\vec{x})$ in the feature layer. Kramer [13] solved this problem by training a multilayer perceptron as shown in Figure 2.1

using backpropagation of error. All of the low-dimensional space representation in this chapter, the feature data are three-dimesional. The feature data were produced by an NLPCA neural network that has three nodes at the feature layer.

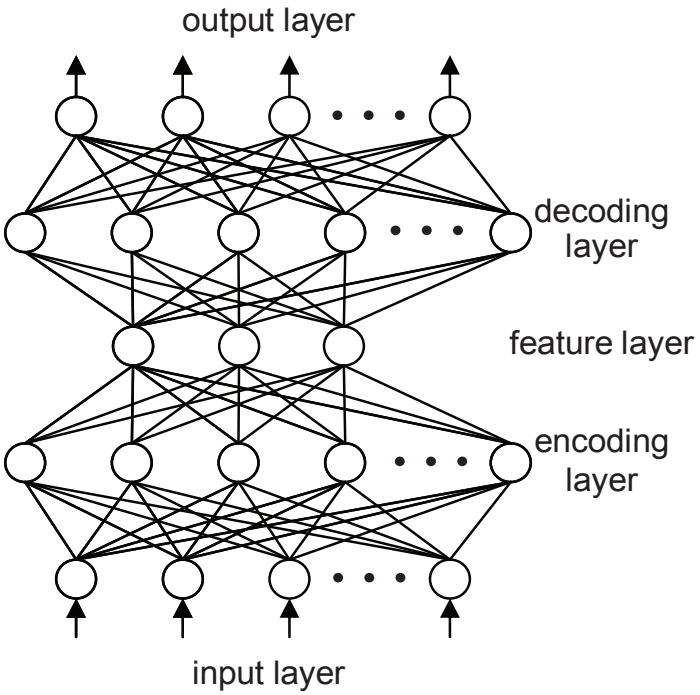

FIGURE 2.1: The structure of the nonlinear principal components network is symmetrical with respect to its feature layer. The number of input nodes and output nodes are set to the number of dimensions of the input data. Target values presented at the output layer are set to be identical to input values. The number of nodes in the encoding layer and the decoding layer increase with the complexity of the data set. Both the encoding layer and decoding layer contain nonlinear nodes. In this work, the number of nodes in the input layer, encoding layer, feature layer, decoding layer, and output layer are 20, 25, 3, 25 and 20, respectively.

PCA is a special case of NLPCA in which $\mathcal{C}$ and $\mathcal{D}$ are linear. A straightforward NLPCA training may not have a unique solution. Correctly setting the initial weights of the NLPCA network is key for convergence. In 2008 Hinton and Salakhutdinov [26] found an effective way to initialize the weights of an NLPCA neural network. Unlike PCA and nonparametric methods such as [27], [28], NLPCA autoencoders give mappings in both directions between high-dimensional space and low-dimensional space.

From preliminary observation of the humanoid motion patterns in the feature space of NLPCA, the patterns are appeared to be closed-curves. This is corresponding with the periodic nature of the data. Conventional NLPCA is unsuitable for learning a closed or self-intersecting curve [29]. However, nonlinear principle component neural networks with a circular constraint at the feature layer (CNLPCA) can overcome this difficulty

[19]. To model these closed-curves, CNLPCA is used for generalization of the periodic motion patterns in the feature space.

Kirby and Miranda [19] constrained the activation values of a pair of nodes p and q in the feature layer of an NLPCA neural network to fall on the unit circle:

$$r = \sqrt{p_0^2 + q_0^2}, \tag{2.2}$$

$$p = \frac{p_0}{r} \text{ and } q = \frac{q_0}{r}. \tag{2.3}$$

While p_0 and q_0 are the input activation, p and q are the output of nodes $\mathbf{p}$ and $\mathbf{q}$, respectively. Thus, the pair of nodes $\mathbf{p}$ and $\mathbf{q}$ act as a single angular variable

$$\theta = \arctan \frac{p}{q}. \tag{2.4}$$

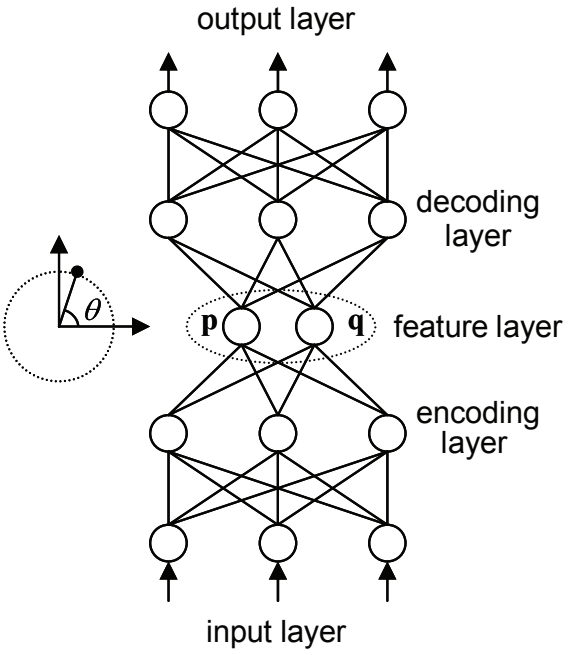

FIGURE 2.2: The NLPCA network with a circular constrain at the bottleneck layer. In this work, the number of nodes in the input layer, encoding layer, feature layer, decoding layer, and output layer of CNLPCA are 3, 3, 2, 3 and 3 respectively as depicted in this figure.

2.2 Automatic segmentation algorithm

Automatic segmentation is this study is conceived as the problem of uniquely assigning a temporal sequence of data points in the feature space to CNLPCA neural networks. As the robot begins to move, the first network is assigned some minimal number of data points, and its training starts with these points. This gets the network learning started quickly and provides it with sufficient information to determine the orientation and curvature of the trajectory. A network accepts points based on its prediction. Once data points from a different pattern are assigned to the learning network, its prediction error rapidly increases, and a new network will be deployed and start learning those data points. The automatic segmentation algorithm works as follows:

TABLE 2.1: Psedocode for automatic segmentation

1.	Initialize a CNLPCA network.
2.	Assign n data points in temporal order to the CNLPCA network.
3.	Let the network learn the assigned data points.
4.	If $MSE_{new} < (1 + \alpha) \times MSE_{old}$ go to step 2.
5.	End learning of this segment.
6.	Go to step 1 until the end of the data set is reached.

From Table 2.1, the automatic segmentation begins to work by deploying a CNLPCA neural network. Then n points of data along the temporal data sequence in low-dimensional space are assigned to the network that was created in the previous step. The value of n is not a critical free-parameter of our algorithm: n could be any positive integer greater than or equal to one. In other words, the parameter n is the size of the new data set that is added to the network to learn a pattern for every iteration of the algorithm. Thus, if we increase n, there will be fewer iterations in Table 2.1. However, value for n should be a fraction of the total number of data points in a segment to avoid biasing the segmentation. After n data points have been assigned to the network, the network training begins. In this work, the terminal criterion of network learning is not the number of epochs. The variables MSE_{new} and MSE_{old} in step 4, are the mean square error values of the learning network at the current step and the previous step, respectively. Step 4 is a crucial step of the automatic segmentation algorithm, because the decision to continue learning on the same segment or to begin learning a new segment is made at this step. The decision is made by comparing the mean square error of the network before and after it has attempted to learn the additional n data points. The second free parameter α is introduced the condition. The parameter α is a small positive real number that is less than one. It indicates how much the mean square error value

of the learning network is permitted to increase when a new set of n data points are assigned to it. This condition usually does not lead to a larger value of the mean square error at the end of segment learning. The mean square error value will decrease again at the next iteration of the learning of the network, if the n newly assigned data belong to the same motion pattern. If learning does not decrease the mean square error, and its value exceeds the condition, the latest n data points will be rejected from the learning segment, and a new segment will begin to learn the n data points. The algorithm keeps deploying CNLPCA neural networks and assigning n data points to them until the end of the data set is reached.

Since the algorithm segments different data patterns in accordance with the temporal constraint of the data set, if there are repeated motion patterns, for example, if the robot walked forward, turned right, and then walked forward again, there will be two segments that represent the walking forward pattern with their corresponding networks. One of these two segments may be considered redundant. One abstract motion pattern should be represented by one network. Thus, the redundant networks should be removed or at least reduced in number. The following steps minimize network redundancy:

TABLE 2.2: Psedocode for network redundancy minimization

1.	For $i = 1, \ldots n$ where n is the total number of segments.
2.	For each segment i, calculate $D_{ij} = \frac{1}{d_{avg}^2}$ to segment j, where $i < j \leq n$, and d_{avg} is an average distance between the two segments.
3.	For all D_{ij}, if D_{ij} exceeds a threshold, merge and relearn the segments that D_{ij} refers to.

To calculate the average distance d_{avg} between segment i and j, one may calculate the average value of the output of network j when the output of network i are given as the input data. The output of network i is obtained by running the angular parameter at the bottleneck layer of the network from 0 to 2π at small increments. The inverse of the square of average distance D_{ij} is used for a clearer discrimination of the distance between segments.

2.3 Automatic motion segmentation of a motion sequence

This section shows the results of automatic segmentation. The accuracy of the results is assessed based on a manual segmentation of the data and an analysis of how data points are allocated among the CNLPCA neural networks. The segmentation results before and after applying network redundancy minimization are also shown.

A motion sequence data were recorded while a human operator manually controlled a Fujitsu HOAP-2 humanoid robot to play soccer, as shown in Figure 2.3. The motion sequences are walking forward, turning right, turning left, walking forward,[1] sidestepping right, sidestepping left, and kicking. Each data point is constituted by a 20-dimension vector of joint angles.[2] After the 20 dimensional joint data were normalized to have zero mean and unity variance, a standard NLPCA network reduced the dimensionality of the data from 20 dimensions to 3 dimensions. The 3-dimensional data results can be visualized in Figure 2.4. These steps are data preprocessing[3] for more efficient automatic segmentation by the CNLPCA algorithm.

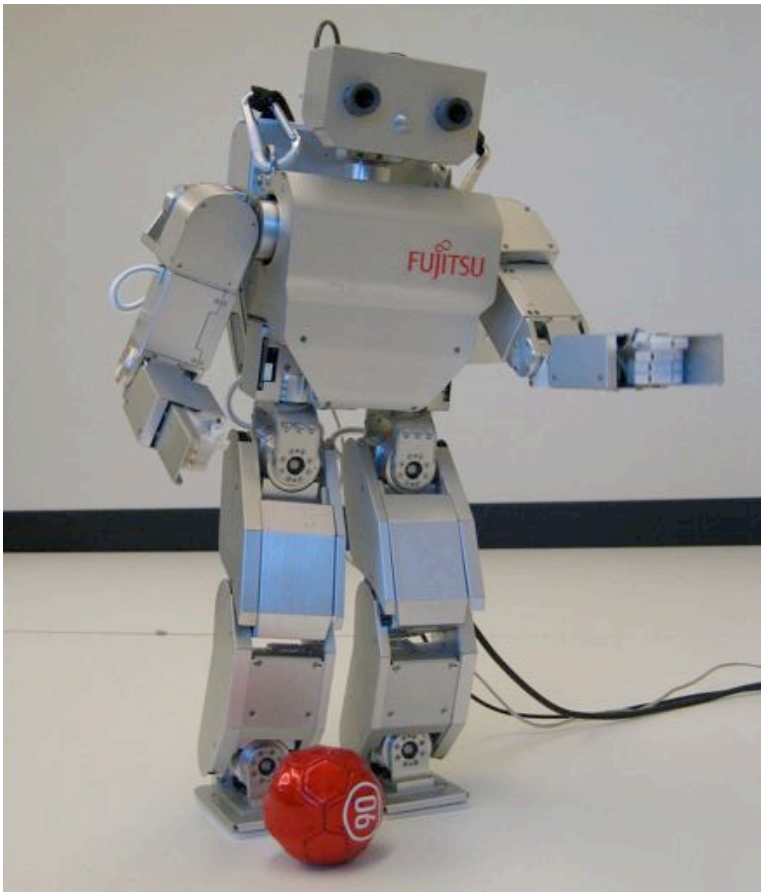

FIGURE 2.3: Fujitsu HOAP-2 robot's ball following behavior. The Fujitsu HOAP2 robot has 25 DOFs: 6 DOFs at each leg, 4 DOFs at each arms, 1 DOFs at each hand and 2 DOFs at the neck.

Eight segments of motion data patterns were classified after the automatic segmentation was performed along the temporal order of the data. An accuracy analysis of the segmentation results is shown in Figure 2.5. The figure compares the average distances

[1] To demonstrate that our algorithm is able to handle redundant motion patterns, the walking forward motion intentionally appears more than one time in the motion sequence.

[2] The Fujitsu HOAP-2 robot has 25 joints, but two neck joints, two hand joints, and one torso joint are not used in motion patterns in this study.

[3] Neural network training can be made more efficient when we perform certain preprocessing steps on the network inputs and targets.

between manually and automatically segmented trajectories. A data points allocation analysis, which indicates the performance of the algorithm at categorizing different patterns of motion data into different segments in the data sequence, is shown in Figure 2.7. After automatic segmentation has completed, the routine for redundant network minimization searches for segments which positions are very close to each other and merges them. Figure 2.4 shows the complete automatic segmentation routine successfully employed CNLPCA neural networks to separate and generalize five of the periodic motions without any prior information about the number or type of motion patterns.

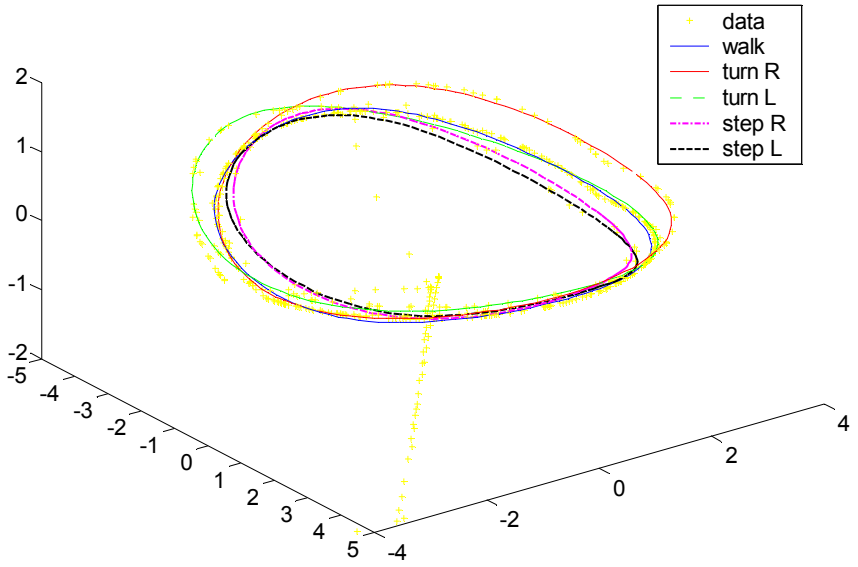

FIGURE 2.4: Recognized motion patterns embedded in the dimensions of the first three nonlinear principal components of the raw proprioceptive data.

The average distance can be calculated from a manually segmented data to an automatically segmented data by providing the points of manually segmented data as input to the target CNLPCA network and calculating the average value of distance between the input and the output. Figure 2.5 and 2.6 are analyses of average distances from each automatically segmented pattern to each manually segmented pattern before and after applying the routine that minimizes redundant segments. There are eight segments in the automatic segmentation results before applying the network redundancy minimization algorithm, as shown in Figure 2.5. The lowest bar indicates which known pattern matches the automatically segmented pattern. We notice from Figure 2.5 that segment No.1, 5, and 8 match the walking pattern. The redundancy among these segments occurred, because the robot performed this action three times during different time intervals when we recorded the data. Thus, this is a correct result of the segmentation algorithm based on the temporal ordering. Segment No. 2 and 3 in Figure 2.5 are also redundant. Both represent the turning right action. This is an inaccurate result,

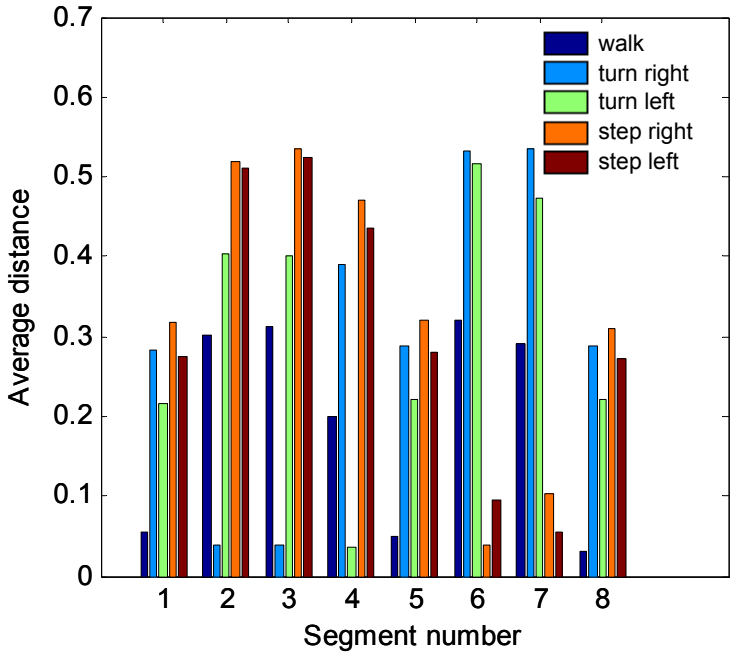

FIGURE 2.5: The average distance between manually and automatically segmented neural networks before eliminating redundant networks. (A shorter bar indicates greater similarity with respect to the reference pattern.)

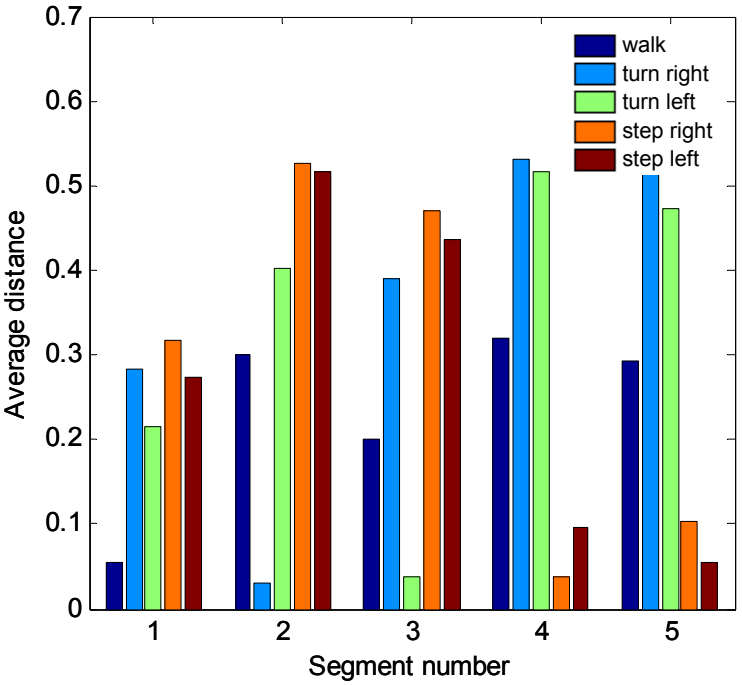

FIGURE 2.6: The average distance between manually and automatically segmented neural networks after eliminating redundant networks. (A shorter bar indicates higher similarity with respect to the reference pattern.)

because the robot performed the turning right action only once during the recording of data.

There should be only one network to represent each motion pattern. Increasing the mean square error value of the learning network parameter α influences the result. The lower the value of α used, the higher the number of likely segments. Although the motion sequence might be divided into several segments at this step, segments that represent the same motion pattern will be merged later by the routine for network redundancy minimization.

Redundant and fragmented segments that represent the same abstract action share similar curvature and lie near each other in the reduced sensorimotor space. The algorithm in Table 2.2 can search and merge these redundant and fragmented segments. All of the redundant networks were removed and their data points were reallocated. Figure 2.7 and 2.8 are an analysis of the allocation of data points before and after applying network redundancy minimization. Each bar represents the percentage of data points that belong to each known pattern in an automatically segmented trajectory. This value is the ratio of the number of data points of each of pattern in a segment to the total number of data points of each pattern in the entire data set. A very low rate of data point misallocation is observed in Figure 2.7. The allocation of data points after the removal of the redundant networks is also accurate. From Figure 2.7, segment No. 5 and 8, which are redundant with respect to segment No. 1, were merged into segment No. 1 in Figure 2.8. Segment No. 3, which is redundant with respect to segment No. 2, was also merged into segment No. 2 in Figure 2.8.

However, this algorithm could not capture the kicking pattern. The kicking is a none-periodic motion. A different functional approximation algorithm that works well with none-periodic motion could be added to the existed algorithm in the future work.

2.4 Summary

In a space of reduced dimensionality, the automatic segmentation algorithm was able to divide sequences of humanoid motion data into segments of periodic motion. The first phase is a temporal ordering segmentation process that combines learning and temporally-constrained data point assignment among multiple neural networks. The second phase is a process of minimizing redundant networks that merges redundant networks based on the average spatial distance between patterns. The automatic segmentation results can be used to facilitate the learning of complex tasks performed by

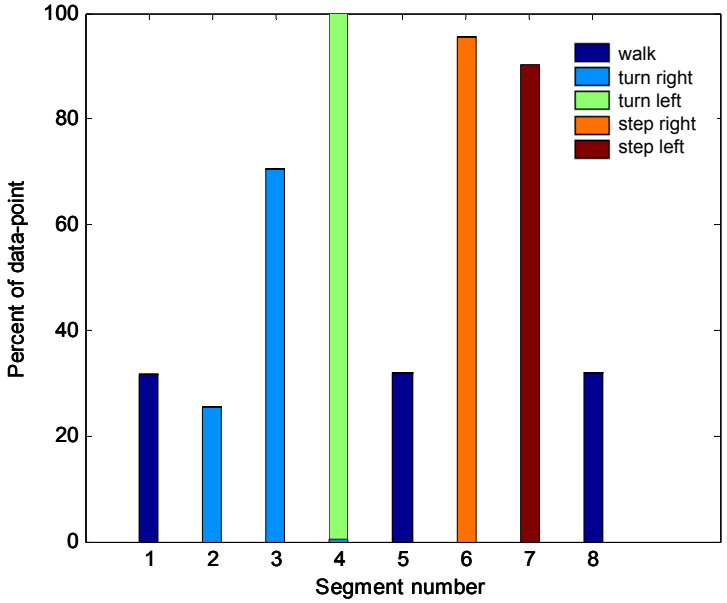

FIGURE 2.7: The allocation of data points to each network before applying network redundancy minimization.

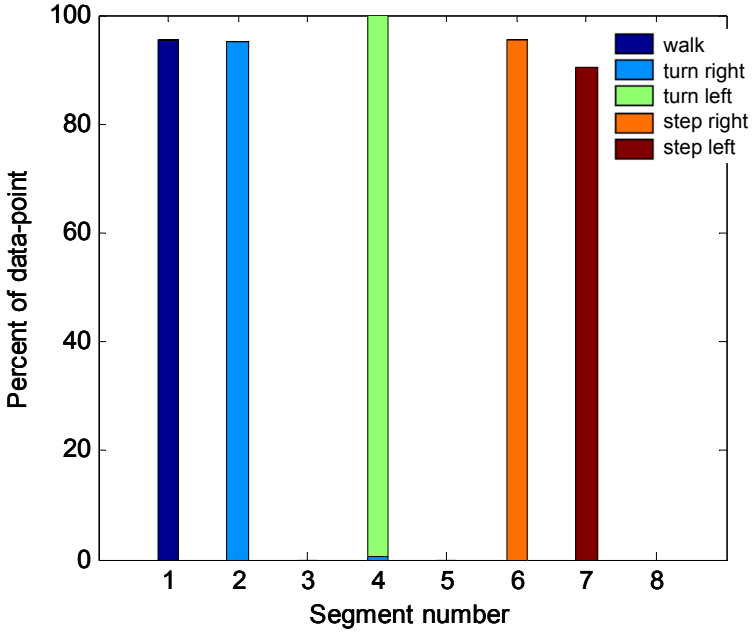

FIGURE 2.8: The allocation of data points to each network after applying network redundancy minimization.

humans by deriving an abstract state-action space for reinforcement learning [24], if dynamically stable motions can be reproduced from the results.

The feature layer of NLPCA provided a low-dimensional compact representation of motion data. By using the decoder part of NLPCA as in Figure 2.1, these 3-D feature data can be used as whole-body joint commands for a humanoid robot. The single angular parameter at the feature layer can also be considered as an extreme dimensionality reduction. The concept of one variable that can govern cycles of motion is very interesting to be investigated further. However, training the NLPCA network takes very long time for a large number of data. Results of NLPCA are also not reliable. Different results can be occurred from different NLPCA networks that were trained from the same data-set. Performance of CNLPCA is good when it is used for modeling planar circular patterns. But, when a closed-curve pattern appears to be an irregular shape, the CNLPCA can not perform well. Even many interesting and useful concepts for motion reproduction have been arisen from the algorithms that were implemented in the study in this chapter, algorithms that have similar properties without the drawbacks that mentioned here should be used, instead.

Chapter 3

The Low-Dimensional Subspaces of Posture

Learning a human motion can be considered as adjusting of whole-body postures. In other words, learning of human motion is generally a problem of whole-body dynamics optimization. Human body has 244 degrees of freedom [25]. The Fujitsu HOAP2 humanoid robot in figure 2.3 has 25 degrees of freedom. As the number of degrees of freedom increases, the system dynamic model become very complex and finding an optimal control policy becomes difficult. Attempting to search for an optimal value in such a high dimensional space is normally a running into the curse of dimensionality problem [30]. However, particular classes of motion such as walking, kicking, or reaching for an object are intrinsically low-dimensional. To overcome the curse of dimensionality problem, a low-dimensional representation of whole-body posture is employed. The compact representation of whole-body posture in low-dimensional subspaces is described in this capter. The compact posture representation is achieved through a linear dimensionality reduction algorithm. This compact posture representation will be later combined with sensory feedback to from an optimal movement policy on the next chapter. The dimensionality reduction method and the feature space, which the compact postures are presence as well as properties of posture in the feature space will be depicted mathematically and graphically in this chapter as well.

3.1 Data pre-processing and PCA mapping

Although, nonlinear dimensionality reduction algorithms have been already applied to representation of human posture such as work in [31] and [32]. However, these methods have some parameters that have to be well-tuned. Properties of the resulted low-dimensional space of these algorithms have not well studied. PCA is a non-parametric linear dimensionality reduction technique. Its algorithm and properties are well studied. For the purpose of unambiguous presentation of the underlying novel ideas of the work in this dissertation, the principal components analysis is employed in the motion learning framework in this dissertation.

Let $\boldsymbol{\theta}$ be a vector whole-body of joint angle data of a humanoid robot that has m joints:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}. \tag{3.1}$$

A motion sequence $\boldsymbol{\Theta}$ of n joint angle vectors can be described in matrix form as:

$$\boldsymbol{\Theta} = \begin{bmatrix} \theta_{11} & \theta_{12} & \dots & \theta_{1n} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{m1} & \theta_{m2} & \dots & \theta_{mn} \end{bmatrix}. \tag{3.2}$$

In figure 3.1, recorded trajectories of joint position from Fujitsu HOAP2 robot that was performing walking motion by a hand-coded program[33] is shown. Note that even HOAP2 robot has 25 joints, for this walking gait only 20 joints are used. Two joints at the neck, one joint at each hand and one joint at torso are not used in this gait. The data were recorded form a standing posture until the robot had walked for five seconds.

A statistical data-preprocessing has to be done before performing PCA mapping. Data of each row of $\boldsymbol{\Theta}$ in equation (3.2) must be normalized such that their mean is zero and their standard deviation is one. Let μ_i be average value (mean) of joint i:

$$\mu_i = \frac{1}{n} \sum_{j=1}^{n} \theta_{ij} \tag{3.3}$$

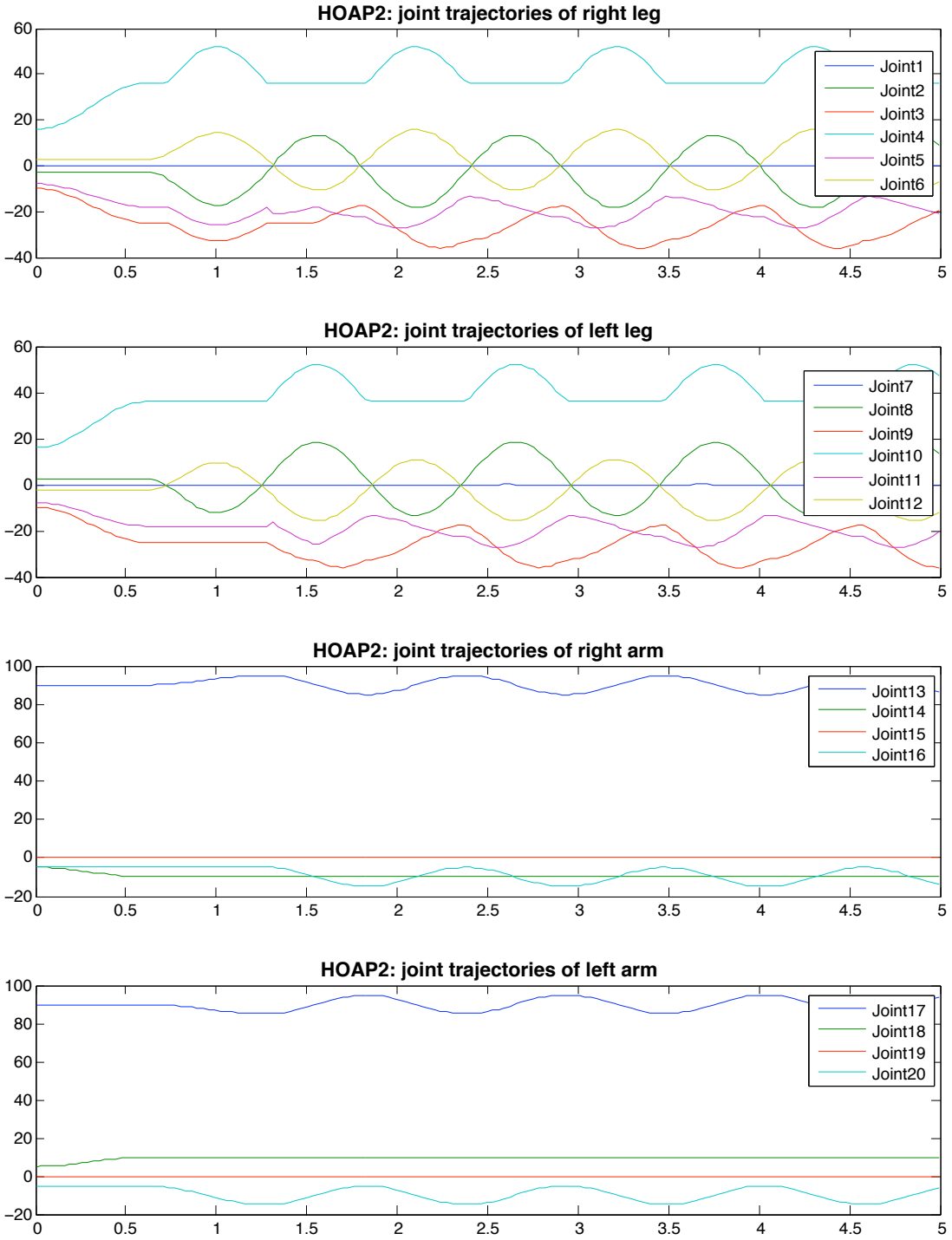

FIGURE 3.1: Joint angle data of a hand-coded walking gait of HOAP2 robot. This is figure, **x** axis is time in second and **y** axis is joint angle in degree.

and σ_i be standard deviation of joint i:

$$\sigma_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(\theta_{ij} - \mu_i)^2}. \tag{3.4}$$

The normalized joint angle data $\mathbf{Q}$ can be derived from $\mathbf{\Theta}$ by,

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ q_{21} & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \dots & q_{mn} \end{bmatrix} \tag{3.5}$$

where

$$q_{ij} = \frac{\theta_{ij} - \mu_i}{\sigma_i} \tag{3.6}$$

when $i = 1 \dots m$ joint and $j = 1 \dots n$ number of data. Let $\mathbf{q_i}$ when $\mathbf{i} = 1 \dots m$ be normalized data of joint $\mathbf{i}$. The normalized data of joint i can be expressed in form of row vector:

$$\mathbf{q_i} = \begin{bmatrix} \mathsf{q}_1 & \mathsf{q}_2 & \dots & \mathsf{q}_n \end{bmatrix}. \tag{3.7}$$

The normalized data $\mathbf{Q}$ can be rewritten in vector form as:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q_1} \\ \mathbf{q_2} \\ \vdots \\ \mathbf{q_m} \end{bmatrix}. \tag{3.8}$$

In figure 3.2, normalized data of joint angle data from figure 3.1 is shown. Notice that data from each joint are already transformed into the same scale of value.

The covariance matrix of $\mathbf{Q}$ can be obtained by:

$$\begin{aligned} \mathbf{A} &= \mathbf{cov}(\mathbf{Q}) \\ &= \begin{bmatrix} \mathbf{cov}(\mathbf{q_1}, \mathbf{q_1}) & \mathbf{cov}(\mathbf{q_1}, \mathbf{q_2}) & \dots & \mathbf{cov}(\mathbf{q_1}, \mathbf{q_m}) \\ \mathbf{cov}(\mathbf{q_2}, \mathbf{q_1}) & \mathbf{cov}(\mathbf{q_2}, \mathbf{q_2}) & \dots & \mathbf{cov}(\mathbf{q_1}, \mathbf{q_1}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{cov}(\mathbf{q_m}, \mathbf{q_1}) & \mathbf{cov}(\mathbf{q_m}, \mathbf{q_2}) & \dots & \mathbf{cov}(\mathbf{q_m}, \mathbf{q_m}) \end{bmatrix} \end{aligned} \tag{3.9}$$

where

$$\mathbf{cov}(\mathbf{q_i}, \mathbf{q_j}) = \frac{\sum_{i=1}^{m}(\mathsf{q}_i - \mu_i)(\mathsf{q}_j - \mu_j)}{m-1}. \tag{3.10}$$

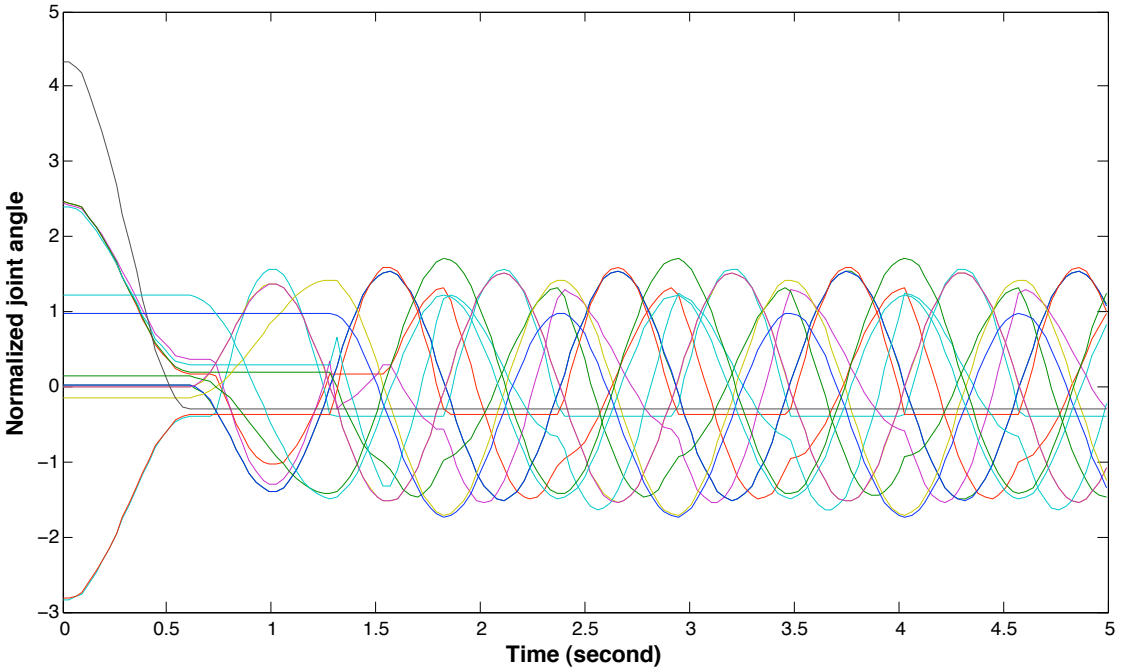

FIGURE 3.2: Normalized joint angle data of the joint angle data in figure 3.1

Since, $\mathbf{A}$ is a square symmetric matrix. Then, eigenvectors of $\mathbf{A}$ can be obtained through performing eigenvalue decomposition. For $i = 1 \ldots m$ each of eigenvalue λ_i, there is a corresponding eigenvector (feature vector) $\mathbf{v}_{\lambda_i}$. To construct a transformation matrix $\mathbf{V}$ for PCA mapping, the eigenvectors $\mathbf{v}_{\lambda_i}$ must be sorted such that $\lambda_1 > \lambda_2 > \ldots \lambda_m$. Thus, the transformation matrix $\mathbf{V}$ can be expressed in term of eigenvectors as:

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_{\lambda_1} & \mathbf{v}_{\lambda_2} & \ldots & \mathbf{v}_{\lambda_m} \end{bmatrix}. \tag{3.11}$$

Let

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \ldots & p_{1n} \\ p_{21} & p_{22} & \ldots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \ldots & p_{mn} \end{bmatrix} \tag{3.12}$$

be the data in the feature space. Once the feature space is formed, the normalized data $\mathbf{Q}$ can be transformed into the feature space by:

$$\mathbf{P} = \mathbf{V}^T \mathbf{Q} \tag{3.13}$$

when $\mathbf{V}^T$ is transpose matrix of $\mathbf{V}$. Each row of $\mathbf{P}$ contains significance of data in decreasing order. To transform the dimensions high-dimensional data $\mathbf{Q}$ to low-dimensional data $\mathbf{X}$, let l be the number of dimensions of $\mathbf{X}$ where $l < m$. The feature data $\mathbf{X}$ can

be expressed in term of submatrix of $\mathbf{P}$ as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{l1} & p_{l2} & \dots & p_{ln} \end{bmatrix}. \tag{3.14}$$

Suppose that three-dimensional data ($l = 3$) in the feature space is the low-dimensional data of interest. The feature data will be:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ x_{31} & x_{32} & \dots & x_{3n} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ p_{31} & p_{32} & \dots & p_{3n} \end{bmatrix}. \tag{3.15}$$

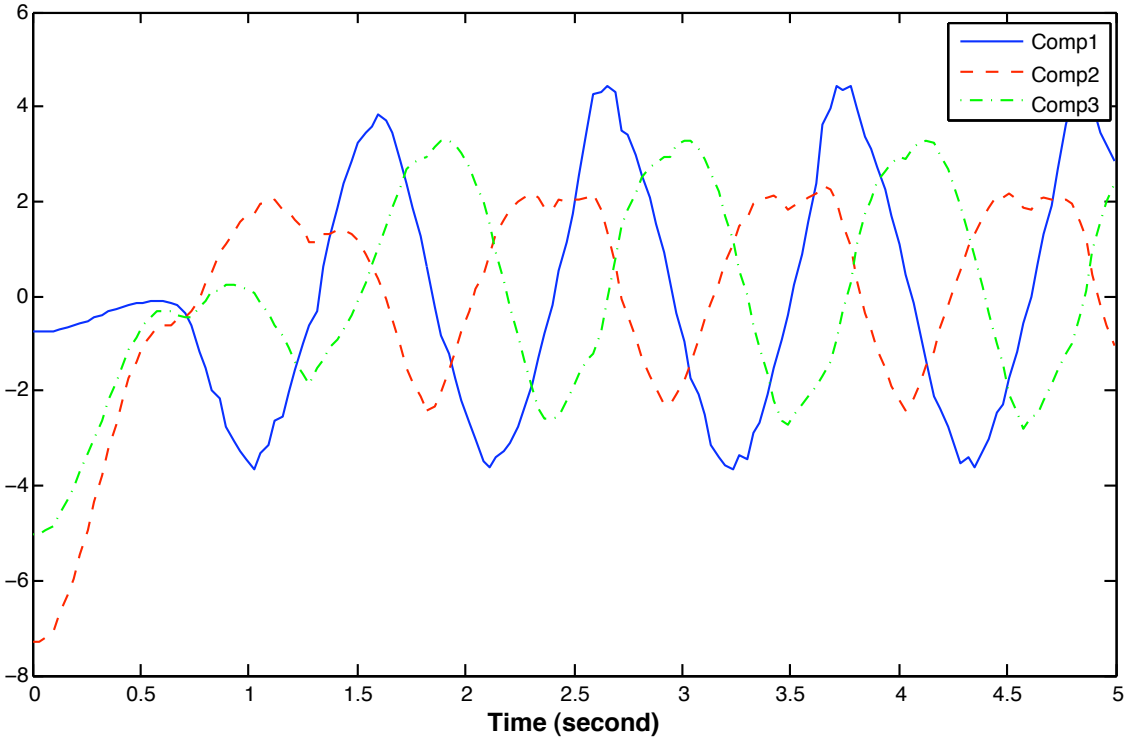

FIGURE 3.3: First three principal components of joint angle data in figure 3.1

Time series of the first three principal components of data is shown in figure 3.3. Notice that variations of data of each principal component are in decreasing order from the first principal to the third principal.

Let inverse mapping data $\widetilde{\mathbf{Q}}$ be an estimation of the normalized data $\mathbf{Q}$. Inverse PCA mapping from the low-dimensional space back to the original high-dimensional space

can be done by:

$$\begin{aligned}\widetilde{\mathbf{Q}} &= (\mathbf{V}^T)^{-1}\widetilde{\mathbf{P}} \\ &= \begin{bmatrix} \tilde{q}_{11} & \tilde{q}_{12} & \dots & \tilde{q}_{1n} \\ \tilde{q}_{21} & \tilde{q}_{22} & \dots & \tilde{q}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{q}_{m1} & \tilde{q}_{m2} & \dots & \tilde{q}_{mn} \end{bmatrix}\end{aligned} \tag{3.16}$$

where

$$\widetilde{\mathbf{P}} = \begin{bmatrix} \mathbf{X} \\ \hline \mathbf{O} \end{bmatrix} \tag{3.17}$$

and

$$\mathbf{O} = \begin{bmatrix} 0_{11} & 0_{12} & \dots & 0_{1n} \\ 0_{21} & 0_{22} & \dots & 0_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{(m-l)1} & 0_{(m-l)2} & \dots & 0_{(m-l)n} \end{bmatrix}. \tag{3.18}$$

The zero matrix $\mathbf{O}$ is combined to the low-dimensional data $\mathbf{X}$ to make the size of matrix $\widetilde{\mathbf{P}}$ to be compatible with $(\mathbf{V}^T)^{-1}$ in equation (3.16). To converse $\widetilde{\mathbf{Q}}$ back to the original data space, a reverse normalization process also has to be performed by:

$$\widetilde{\mathbf{\Theta}} = \begin{bmatrix} \tilde{\theta}_{11} & \tilde{\theta}_{12} & \dots & \tilde{\theta}_{1n} \\ \tilde{\theta}_{21} & \tilde{\theta}_{22} & \dots & \tilde{\theta}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\theta}_{m1} & \tilde{\theta}_{m2} & \dots & \tilde{\theta}_{mn} \end{bmatrix} \tag{3.19}$$

where

$$\tilde{\theta}_{ij} = \tilde{q}_{ij}\sigma_i + \mu_i. \tag{3.20}$$

In the case of data in the feature space $\mathbf{X}$ has the same dimension with $\mathbf{Q}$ or in the case of $l = m$.

$$\widetilde{\mathbf{\Theta}} = \mathbf{\Theta} \tag{3.21}$$

will be the result. If $l < m$, some accuracy of the data will be lost. Figure 3.4 shows accuracy accumulation along numbers of principal axes of PCA mapping of the data from figure 3.1. For example, for using three-dimensional data in the feature space, only about 88% of data accuracy can be recovered after direct and inverse PCA mapping. And if one would like to be able to recover 100% of accuracy of data from PCA mapping, eight-dimensional data in the feature space have to be used.

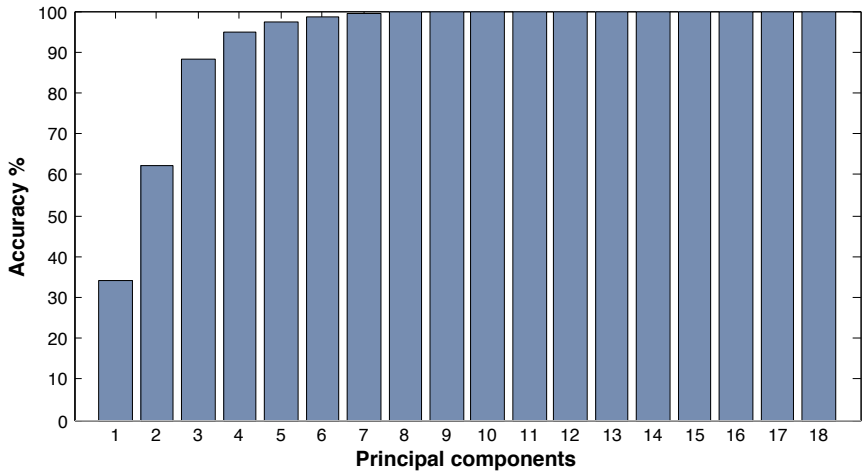

FIGURE 3.4: Accuracy accumulation on principal components for hand-coded walking motion data from figure 3.1

.

3.2 Low-Dimensional Representation of Postures

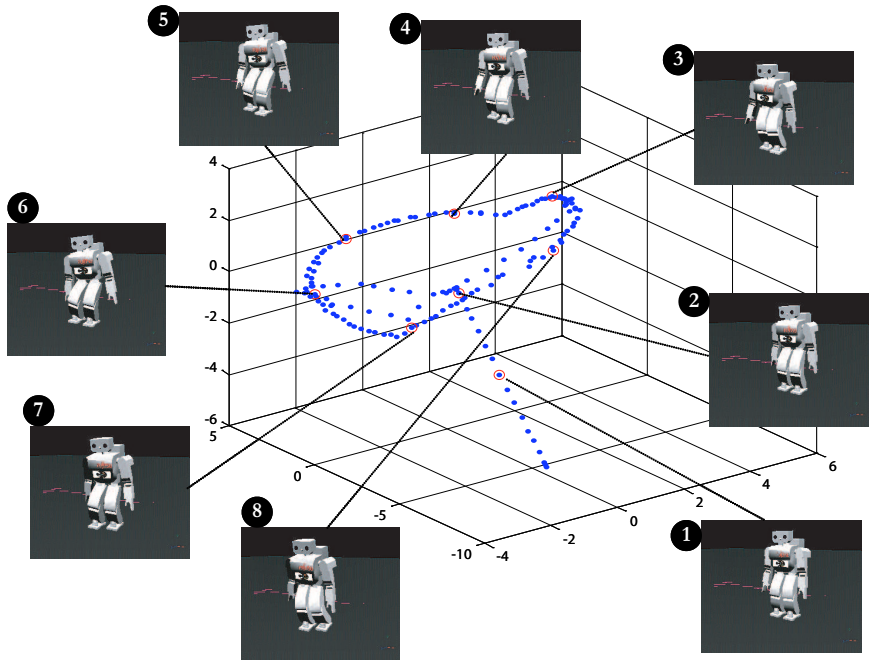

FIGURE 3.5: Posture subspace and example poses from a hand coded walking gait. A three-dimensional space produced by PCA represents the posture of the Fujitsu HOAP2 robot. Blue points along a loop represent different robot postures during a single walk cycle. The first two labeled postures are intermediate postures between an initial stable standing pose and a point along the periodic gait loop represented by postures three through eight.

Principal components analysis forms the low-dimensional motion subspace $\mathbb{X}$. Vectors of joint angle data in the high-dimensional space are mapped to the low-dimensional space by multiplication with the transformation matrix $\mathbf{V}^T$. The columns of $\mathbf{V}$ consist of

the eigenvectors, computed via eigenvalue decomposition, of the motion data covariance matrix. Eigenvalue decomposition produces transformed vectors whose components are uncorrelated and ordered according to the magnitude of their variance. These transformed vectors shall be referred as *eigenposes*.

An example of the three-dimensional representation whole-body posture data (3-D eigenpose data) in the feature space of the HOAP-2 robot executing a walking gait of the data in figure 3.1 are shown in figure 3.5. Data from the first, second and third principal components are plotted on **x**, **y** and **z** axes, respectively. Small robot pictures labeled in the figure are produced by inverse PCA mapping. From this figure, notice that the temporal sequence of motion data is still preserved in the low-dimensional space. As mentioned previously at the end of section 3.1 that some accuracy is lost when three-dimensional feature data are used, and eight-dimensional feature data have to be used for recovery of 100% accuracy of joint angle data. Computation complexity is generally increased exponentially with number of dimensions. Using high number of dimensions of data could lead to the curse of dimensionality problem and break the purpose of using dimensionality reduction. However, the goal of motion optimization or goal-based learning by imitation is not mimicking exact kinematic motion. Furthermore, because of differences in dynamic properties between two robot bodies or human and robot mimicking the same exact kinematic motion while maintaining in a dynamically stable condition may not be possible. For dimensionality reduction through PCA, higher number of dimensions in the feature space can achieve higher accuracy of the data. Three dimensional data are convenient for visualizing and developing motion optimization algorithm through analytical geometry. In this dissertation, low-dimensional data have three dimensions unless state otherwise. Physical meaning or interpretation of parameters in the feature space will be explained along with simulation and experimental results in Chapter 4.

3.3 Action Subspace Embedding

The redundancy of posture data in high dimensional joint space space has been eliminated by PCA mapping. The reduced dimensional subspace $\mathbb{X}$ is used for constraining postures of a motion pattern.

A periodic movement such as walking can be represented by a closed-curve pattern $\mathbb{X}$. The periodic part of the data in Figure 3.5 was manually segmented. The blue dots pattern in Figure 3.6 is the periodic segment of the walking data in Figure 3.5. In the general case, we consider a non-linear manifold representing the action space $\mathbf{A} \subseteq \mathbb{X}$. Non-linear parameterization of the action space allows further reduction in dimensionality. A one-dimensional representation of the original motion in the three dimensional

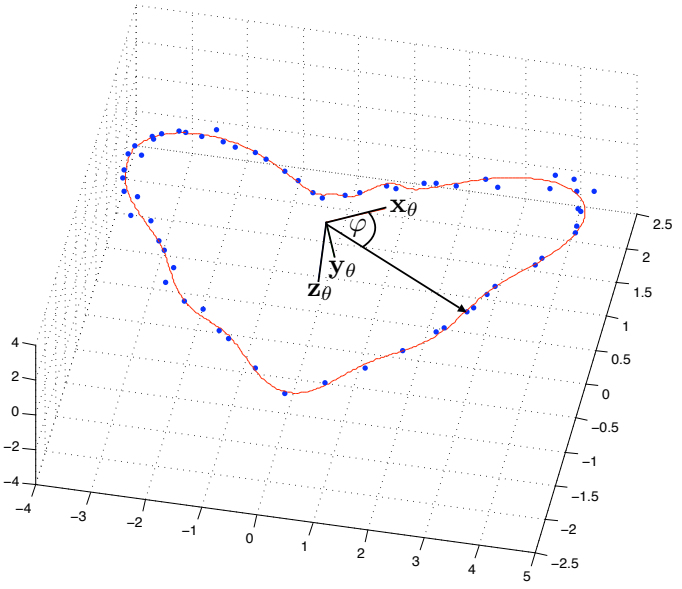

FIGURE 3.6: Embedded action subspace of a humanoid walking gait. Training data points in the reduced posture space (shown in blue-dots) are converted to a cylindrical coordinate frame relative to the coordinate frame $\mathbf{x}_\theta, \mathbf{y}_\theta, \mathbf{z}_\theta$. The points are then represented as a function of the phase angle φ, which forms an embedded action subspace (shown in red solid-line curve).

feature space is embedded. It is used for constructing a constrained search space for optimization which will be discussed in Section 4.2. Using the feature representation of the set of initial training examples $\mathbf{x}^i \subseteq \mathbb{X}$, we first convert each point to its representation in a cylindrical coordinate frame. This is done by establishing a coordinate frame with three basis directions $\mathbf{x}_\theta, \mathbf{y}_\theta, \mathbf{z}_\theta$ in the feature space. The zero point of the coordinate frame is the empirical mean of the data points in the reduced space. The data are re-centered around this new zero point and denote the resulting data $\hat{\mathbf{x}}^i$.

Then, the principal axis of rotation $\mathbf{z}_\theta$ is computed:

$$\mathbf{z}_\theta = \frac{\Sigma_i(\hat{\mathbf{x}}^i \times \hat{\mathbf{x}}^{i+1})}{\|\Sigma_i(\hat{\mathbf{x}}^i \times \hat{\mathbf{x}}^{i+1})\|} \tag{3.22}$$

Next, $\mathbf{x}_\theta$ is chosen to align with the maximal variance of $\mathbf{x}^i$ in a plane orthogonal to $\mathbf{z}_\theta$. Finally, $\mathbf{y}_\theta$ is specified as orthogonal to $\mathbf{x}_\theta$ and $\mathbf{z}_\theta$. The final embedded training data is obtained by cylindrical conversion to (φ, r, h) where r is the radial distance, h is the height above the $\mathbf{x}_\theta - \mathbf{y}_\theta$, and φ is the angle in $\mathbf{x}_\theta - \mathbf{y}_\theta$ plane measured counter-clockwise from $\mathbf{x}_\theta$.

Given the loop topology of the latent training points, one can parameterize r and h as a function of φ. The embedded action space is represented by a learned approximation of the function:

$$[r, h] = g(\varphi) \tag{3.23}$$

where $0 \leq \varphi \leq 2\pi$. Approximation of this function is performed by using a radial basis function (RBF) network. The angle φ can also be interpreted as the motion phase angle. Since, when φ sweeps from 0 to 2π, it creates an action subspace $\mathbf{A}$ which in our case is a walking gait. The parameter φ indicates how far the current posture is from the beginning of the motion cycle. The first order time derivative of φ also tells us the speed of movement.

3.4 Action Subspace Scaling

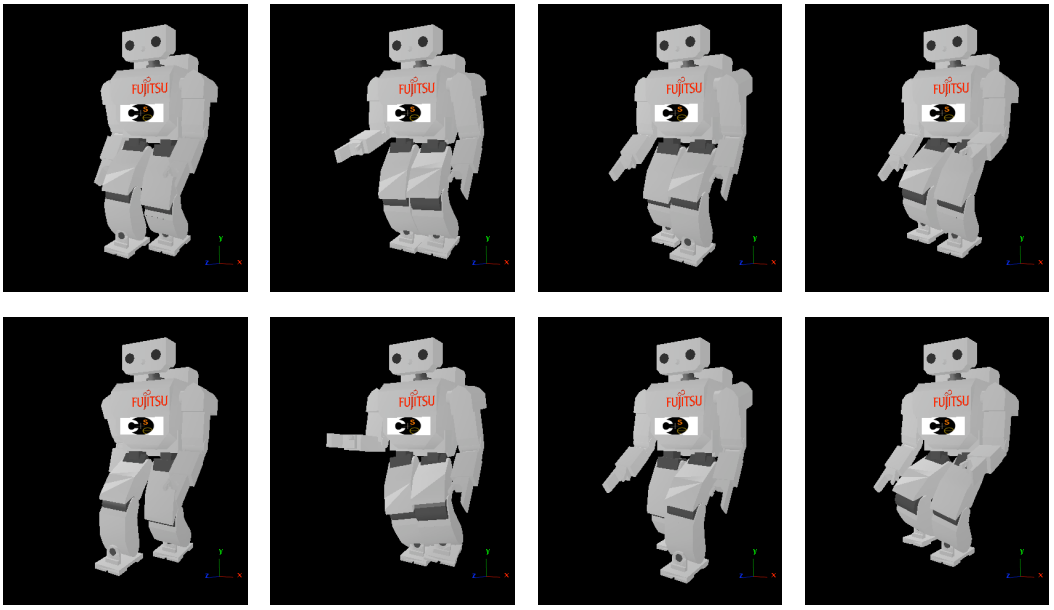

FIGURE 3.7: Motion scaling of a walking gait. The first row of this figure shows four different postures of a walking gait. The second row shows coherent postures of the first row when a multiplying factor $f = 2.0$ is applying to the low-dimensional representation of this walking gait.

As described in section 3.1, high-dimensional joint angle data are normalized before they are processed by a dimensional reduction algorithm. The data among each joint (each dimension) originally are in different scales of values, but after normalization they are scaled into the same range. When the normalized data is multiplied by a scalar value, the results are similar postures with a different magnitude. However, multiplying a vector of raw joint angles data by a scalar factor does not yield a similar posture. A condition for producing a scaled similar posture with respect to a particular class of motion is, the scaling factor must be contributed differently on each joint based on proportion of

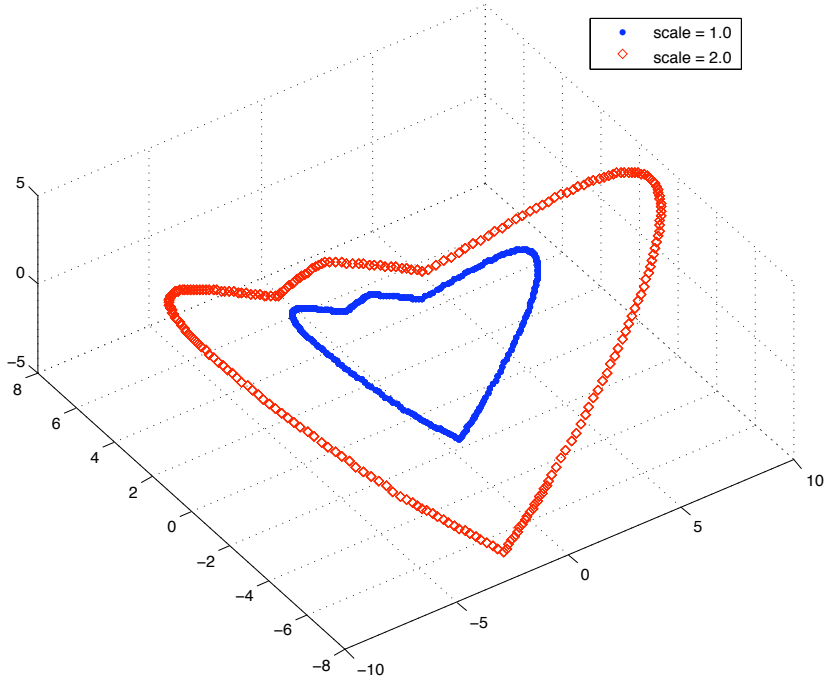

FIGURE 3.8: Corresponding low-dimensional posture data representation of Figure 3.7. The blue dot makers represent a walking gait of Fujitsu HOAP-2 robot in three-dimensional feature space at scale = 1.0. The red diamond makers represent the same data pattern with a multiplying scale 2.0.

motion-range of that joint with respect to the motion. The inverse normalization process implicitly creates this condition. Thus, we may conclude that the normalization process, leads to posture scaling ability. The posture scaling yields reasonable results only when the motion data set contains only one specific type of motion.

From studying of four different motion patterns in the low-dimensional subspace, scaling up and down the patterns produce similar motion patterns with differences in the magnitude of motion. This means posture scaling ability is preserved after PCA is applied. Thus, multiplication of a scalar value to the action space $\mathbf{A} \subseteq \mathbb{X}$ yields a similar action. If $\mathbf{A}$ represents a walking gait, multiplying $\mathbf{A}$ by a factor $f > 1$ will result in a similar walking gait but with a larger step. Multiplying $\mathbf{A}$ by a factor $f < 1$ results in a walking gait with a smaller step size. Note that scaling of an action space is always performed with respect to the mean value of $\mathbf{A}$ or the origin of the cylindrical coordinate frame $(\mathbf{x}_\theta, \mathbf{y}_\theta, \mathbf{z}_\theta)$. In figure 3.7, an example of postures scaling of a walking motion is shown. And the corresponding three-dimensional data in the feature space, which the posture are created from are shown in figure 3.8.

However, action subspace scaling only produces similarity of kinematic postures. The result of scaling may not be dynamically stable, especially when the scaling factor $f > 1$. To achieve stable motion, the new motion has to be gradually learned as described in

chapter 4. Action subspace scaling for $f < 1$ will used for initializing the learning process in learning through imitation in chapter 5.

3.5 Summary

In Chapter 3 and the rest of this dissertation, the low-dimensional subspaces are created by linear PCA. Humanoid motion data in three-dimensional subspaces are defined as *3-D eigenpose data*. Data preprocessing, PCA transformation, inverse PCA transformation, action subspace embedding and action subspace scaling are described in details in this chapter. The action subspace embedding is a set of eigenpose data that embed in the low-dimensional spaces. By using action subspace embedding, a complete motion cycle of a periodic motion can be generated from a single parameter function in Equation 3.23 by varying φ from 0 to 2π. This function is also used for constructing the search-space for motion optimization in later chapters. The action subspace embedding was designed imitates a characteristic of CNLPCA for having a single angular parameter that can reproduce a periodic motion cycle. The action subspace scaling is scaling the size action subspace embedding in the low-dimensional subspaces, which creates a similar motion with different scale of magnitude of movement. This property is used for creating a smaller scale movement of a motion, which is more dynamically stable.

Chapter 4

Learning and Prediction

The methodologies for representing humanoid motion in low-dimensional subspaces have been developed in chapter 3. In this chapter, the 3-D eigenposes are treated as motion command or *action*. The *action* will be combined with sensory feedback or *state* to from a Markovian predictive model. Then, the predictive model will be used for deriving optimal motion commands or *action plan* based on a constraint from action subspace embedding function to acheive dynamically stable motion.

4.1 Learning to Predict Sensory Consequences of Actions

A key component of methodology of this dissertation is learning to predict future sensory inputs based on actions. This learned predictive model is used for optimal action selection. The goal is to predict, at time step t, the future sensory state of the robot, denoted by s_{t+1}. In general, the state space $\mathbf{S} = \mathbf{\Theta} \times \mathbf{P}$ is the Cartesian product of the high-dimensional joint space $\mathbf{\Theta}$ and the space of other percepts $\mathbf{P}$. Other percepts could include, for example, measurements of from torso accelerometer or gyroscope, foot pressure sensors as well as information from camera images. The goal then is to learn a more compact function $F : \mathbf{S} \times \mathbf{A} \mapsto \mathbf{S}$ that maps the current state and action to the next state. For this dissertation, F is assumed to be deterministic.

Often the perceptual state s_t is not sufficient for predicting future states. In such cases, one may learn a higher order mapping based on a history of perceptual states and actions, as given by an n-th order Markovian function:

$$s_{t+1} = F(s_t, s_{t-1}, ..., s_{t-n-1}, a_t, a_{t-1}, ..., a_{t-n-1}) \tag{4.1}$$

In this dissertation unless state otherwise, the time-delay RBF network is used for approximation of predictive sensory-motor model F. In general case, the RBF network approximates F by learning a function $F'(\alpha) = \beta$, when:

$$\beta = \sum_{k}^{K} \mathbf{w}_k \exp(-(\alpha - \mu_k)^T \Sigma_k^{-1} (\alpha - \mu_k)), \tag{4.2}$$

where K represents the number of kernels, μ_k and Σ_k^{-1} are the mean and inverse covariance of the k-th kernel respectively. The output weight vector $\mathbf{w}_k$ scales the output of each kernel appropriately, and the input and output are

$$\alpha = [s_t, s_{t-1}, \ldots, s_{t-n-1}, a_t, a_{t-1}, \ldots, a_{t-n-1}] \tag{4.3}$$

and

$$\beta \approx s_{t+1} \tag{4.4}$$

respectively. Note that the above RBF network can be viewed as a time-delay recurrent network. The history of previous states and actions is implicitly remembered by the network. In this work, a second-order ($n = 2$) time-delay RBF network is used, where the state vector is the three-dimensional gyroscope signal ($s_t \equiv \omega_t$). As discussed in the previous section, an action is represented by a phase angle, radius, and height in latent posture space ($a_t \equiv \chi_t \in \mathbb{X}$). A schematic diagram of predictor learning is illustrated in Figure 4.1. Predicted gyroscope data versus actual gyroscope data during a motion test sequence are shown in Figure 4.2. Notice in Figure 4.2 that the predictor has delivered good prediction of gyroscope signals.

4.2 Motion Optimization using the Learned Predictive Model

The algorithm that is presented in this section utilizes optimization concept and sensory prediction from the previous section to select optimal an action plan for a humanoid robot in a closed-loop feedback scenario. Figure 4.3 illustrates the optimization process.

One may express the desired sensory states that the robot should attain during a particular class of action through an objective function $\Gamma(\mathbf{s})$. The algorithm then selects actions $a_t^*, \ldots, a_T^*$ such that the predicted future states $s_t, \ldots, s_T$ will be optimal with respect to $\Gamma(\mathbf{s})$:

$$a_t^* = \arg\min_{a_t} \Gamma(F(s_t, \ldots, s_{t-n-1}, a_t, \ldots, a_{t-n-1})). \tag{4.5}$$

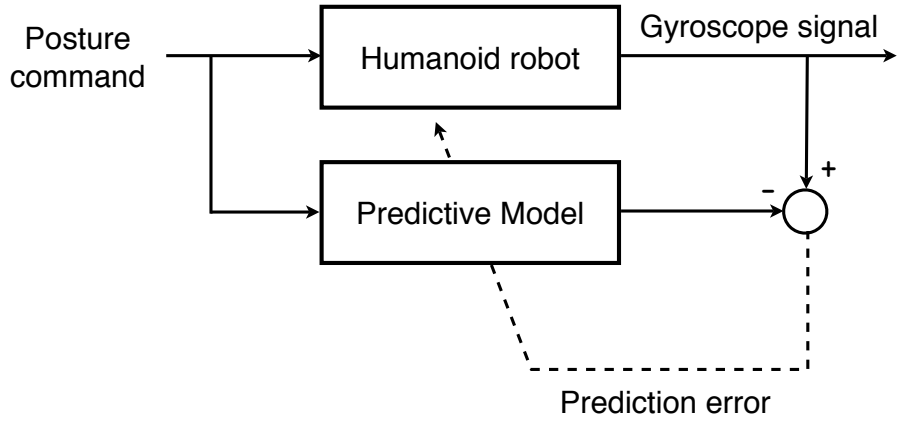

FIGURE 4.1: Sensory-motor stability prediction module. Motion stability (as measured by a three-channel gyroscope sensor at the center of the torso) is predicted based on the input of posture command in the low dimensional space and information of both gyroscope signals and posture command from previous time steps. The predictor network is trained by comparing the predicted gyroscope signals to the actual sensor reading from the robot.

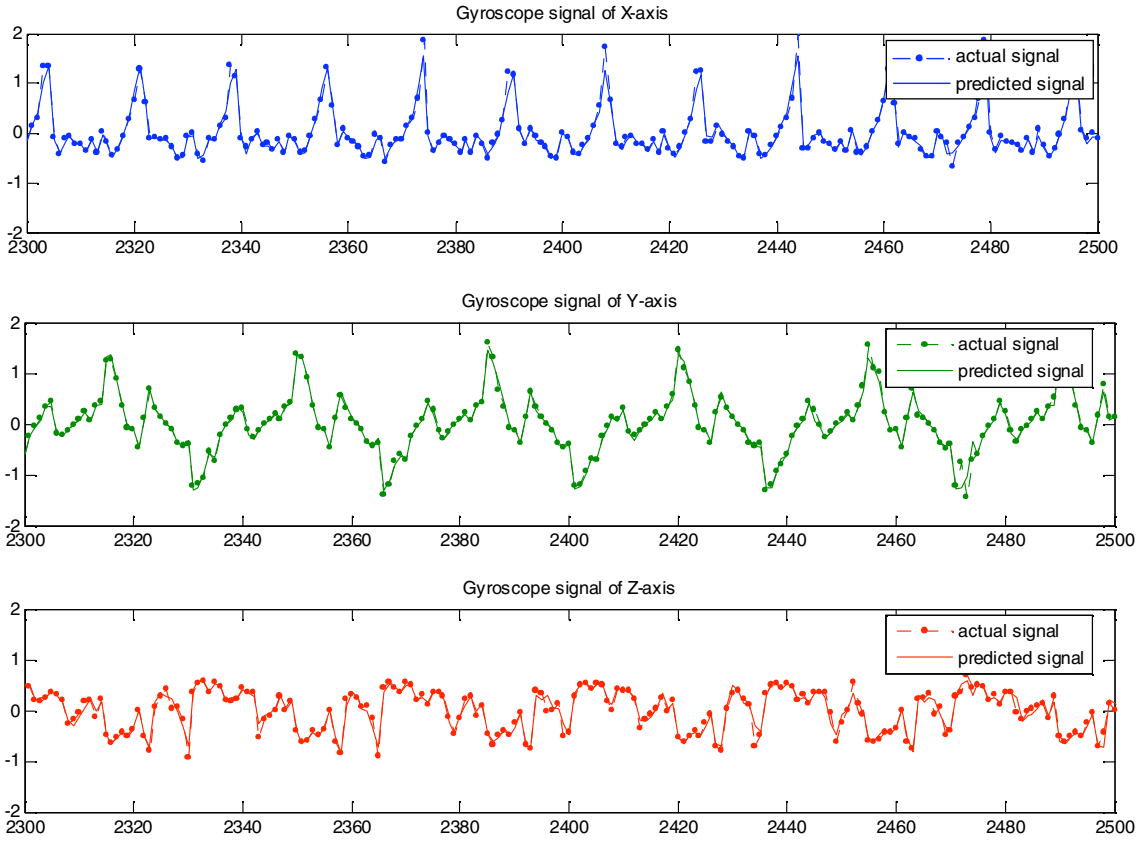

FIGURE 4.2: Gyroscope signal prediction. A second-order time-delay radial basis function network is able to accurately predict gyroscope signals at the next time step. The plots from top to bottom represent individual gyroscope signals $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ during many periods of walking simulation.

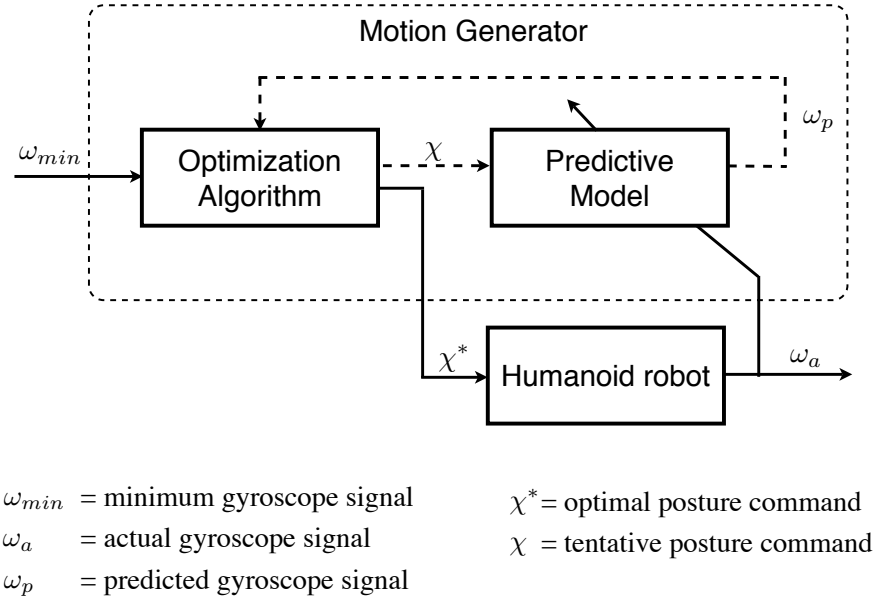

FIGURE 4.3: Model predictive motion generator for optimizing motion stability. At time t, the optimization algorithm generates tentative actions or posture commands ($a_t \equiv \chi \in \mathbb{X}$). The predictive model predicts values of subsequence gyroscope signal ω_p. The optimization algorithm then selects the optimal posture command χ^* based on that satisfies the objective condition ω_{min} based on ω_p. The optimal posture command χ^* is sent to the execute on a robot/simulator. The actual values subsequence gyroscope signal are recorded for retraining of the predictive model.

The objective function used measurement of torso stability as defined by the following function of gyroscope signals:

$$\Gamma(\boldsymbol{\omega}) = \lambda_x \omega_x^2 + \lambda_y \omega_y^2 + \lambda_z \omega_z^2, \tag{4.6}$$

where $\omega_x, \omega_y, \omega_z$ refer to gyroscope signals in the $\mathbf{x}, \mathbf{y}, \mathbf{z}$ axes respectively. The constants $\lambda_x, \lambda_y, \lambda_z$ allow one to weight rotation in each axis differently. Assuming that the starting posture is statically stable, one may simply minimize overall rotation of the robot body during the motion to maintain balance by minimizing sum of square of gyroscope signals. Thus, the objective function (4.6) provides a measure of stability of the posture during motion. For the second-order predictive function F, the optimization problem becomes one of searching for the optimal stable action at time t given by:

$$\chi_t^* = \arg\min_{\chi_t \in S} \Gamma(F(\omega_t, \omega_{t-1}, \chi_t, \chi_{t-1})) \tag{4.7}$$

$$S = \left\{ \begin{bmatrix} \varphi_s & r_s & h_s \end{bmatrix}^T | \varphi, r, h \text{ defined in Section 3.3} \right\} \tag{4.8}$$

To allow for efficient optimization, the search space is restricted to a local region in the action subspace as given by:

$$\varphi_{t-1} < \varphi_s \leq \varphi_{t-1} + \epsilon_\varphi \tag{4.9}$$

$$r_a - \epsilon_r \leq r_s \leq r_a + \epsilon_r \tag{4.10}$$

$$h_a - \epsilon_h \leq h_s \leq h_a + \epsilon_h \tag{4.11}$$

$$0 < \epsilon_\varphi < 2\pi \tag{4.12}$$

$$[r_a, h_a] = g(\varphi_s) \tag{4.13}$$

The phase-motion-command search-range φ_s begins after the position of the phase motion command at the previous time step φ_{t-1}. The radius search r_s range begins from a point in the action subspace embedding $\mathbf{A}$ that is defined by (4.13) in both positive and negative directions from r_a along $\mathbf{r}$ for the distance $\epsilon_r \geq 0$. The search range h_s is defined in the same manner as r_s according to h_a and ϵ_h. In the experiments, the parameters $\epsilon_\varphi, \epsilon_r$,and ϵ_h were chosen to ensure efficiency while at the same time allowing a reasonable range for searching for stable postures. A graphical illustration of the search space for a walking motion is shown in Figure 4.4[1]. For optimization of one cycle walking gait, the search space will move along the constraint function in (3.23) as vaule of φ is increased.

Selected actions will only truly be optimal if the sensory-motor predictor is accurate. Therefore, the prediction model is periodically re-trained based on the posture commands generated by the optimization algorithm and the sensory feedback obtained from executing these commands. After training collectively of three iterations of sensory-motor prediction learning, an improved dynamically balanced walking gait is obtained. Figure 4.4 shows a trajectory of the optimized walking gait in the low dimensional subspace. Note here that the constraint function (3.23) is adapted for every learning iteration. As a result, the search space, which builds around the constraint function are

[1]The search space in this Figure 4.4 is only for graphical illustration purpose. It is not an actual search space.

changed for every learning iteration. The optimization in this paper is a straightforward brute-force search.

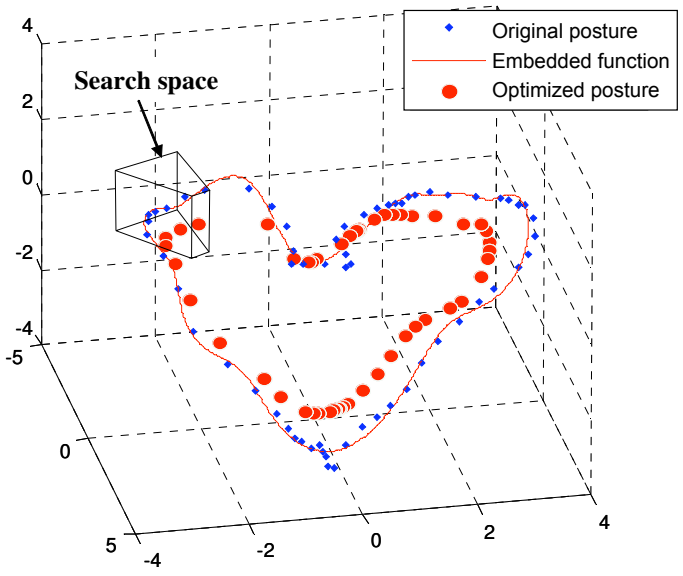

FIGURE 4.4: Optimization result for a walking motion pattern in a low-dimensional subspace based on an action subspace embedding.

The entire motion optimization and action selection process can be summarized below:

1. Use PCA to obtain eigenpose data from the joint data.
2. Apply the action subspace embedding convention for parameterization of the periodic motion pattern.
3. Start the learning process by inverse mapping the eigenpose actions back to the original joint space and executing the corresponding sequence of servo motor commands in a simulator or a real robot.
4. Use the sensory and motor inputs from the previous step to update the sensorymotor predictor as described in Section 4.1. In this chapter, the state vector is comprised of three channels of the gyroscope signal and the action variables are φ, r and h in the low-dimensional subspace.
5. Use the learned model to estimate sequence of actions according to the model predictive controller scheme described above (Figure 4.3).
6. Execute computed actions and record sensory (gyroscope) feedback.
7. Repeat steps 4 through 6 until a satisfactory motion is obtained.

4.3 One-dimensional Optimization

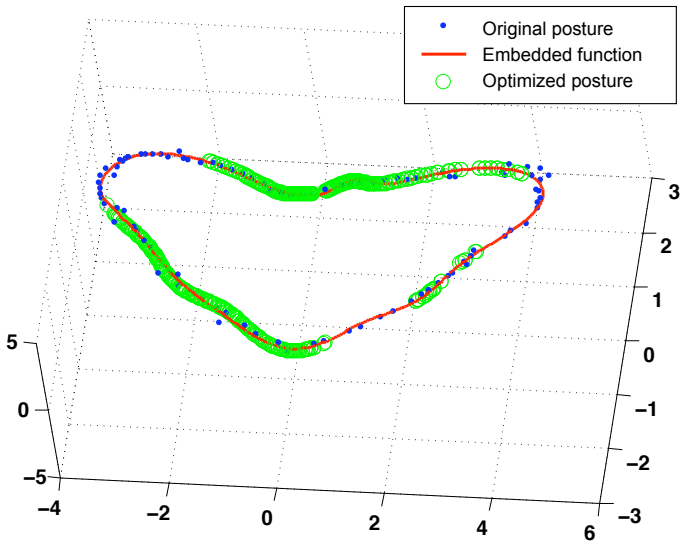

FIGURE 4.5: Motion-phase optimization.

The first experiment is simulation in Webots dynamic environment [34]. And, a real robot can be interchanged with the simulator. This simulation is an experiment to increase the stability of a regenerated hand-coded walking gait shown in Figure 3.5 by using the motion optimization technique in the feature space. This experiment also demonstrates the utility of action subspace embedding and the physical meaning of the parameter φ. Since this experiment is one-dimensional optimization the parameters ϵ_r and ϵ_h in equations 4.10 and 4.11 are set to zero. Then, equation 4.7 becomes:

$$\varphi_t^* = \arg\min_{\varphi_t} \Gamma(F(\omega_t, \omega_{t-1}, \varphi_t, \varphi_{t-1})). \tag{4.14}$$

This process can be referred as *motion-phase optimization*. Because, only the parameter φ is optimized while values of r and h are implicitly optimized through equation 3.23. At the first learning episode, joint angle data that are approximated from an inverse PCA mapping from the three-dimensional feature data in Figure 3.6. Then, their subsequent gyroscope signals were recoded. In this experiment according to equation 4.14, the three channels gyroscope signals are regarded as *state* and the φ is regarded as *action*. These *state-action* data then were used for the time-delay RBF network (depicted in figure 4.1) to learn a predictive model of the gyroscope signals at the next time step. The optimization algorithm used the predictor to obtain a new optimized action plan. The algorithm samples points in the search space defined by equation 4.9 uniformly. Values of gyroscope signals of the sampled points are then calculated. An optimized action command of the action plan is found by selecting a point that gives the minimal sum

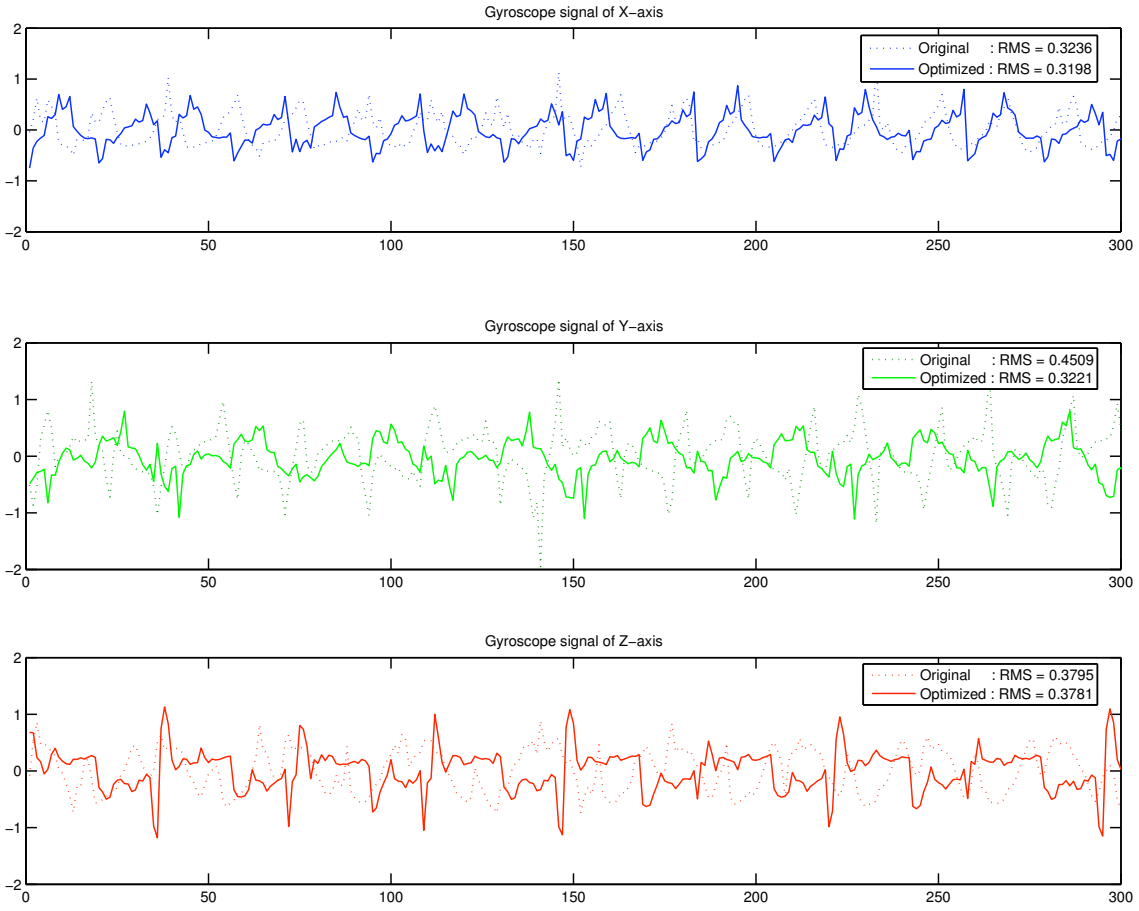

FIGURE 4.6: The plots from top to bottom show the gyroscope signals for the axes X,Y, and Z recorded during initial and optimized walking motions. Root mean squared (RMS) of the gyroscope readings are also indicated in the plot legends. Notice that for the **y**-axis (vertical direction) the RMS values are significantly reduced for the optimized motion.

of square of the gyroscope signals according the the objective function in equation 4.6. The algorithm iteratively calculate another optimal point based on the previous action command and predicted gyroscope signals until the motion cycle is completed. Then, the optimal action plan is executed in the simulator.

The optimization result after three episodes of learning is shown in Figure 4.5. As shown in the figure, motion-phase optimization is especially a line-search. The result of the optimization remains on the constraint pattern. Thus, no new posture is derived from this optimization. However, the phase of the motion is altered in an optimal way of learning based on sensory feedback. The algorithm selects actions that minimize gyroscope signal oscillation. The plots of gyroscope signals and their root mean square (RMS) values for the original walking gait and the optimized one are shown in Figure 4.6. The RMS values of the **y**-axis gyroscope (vertical direction) for the optimized walking gait are significantly lower than for the original one (0.3221 vs 0.4509 respectively). This

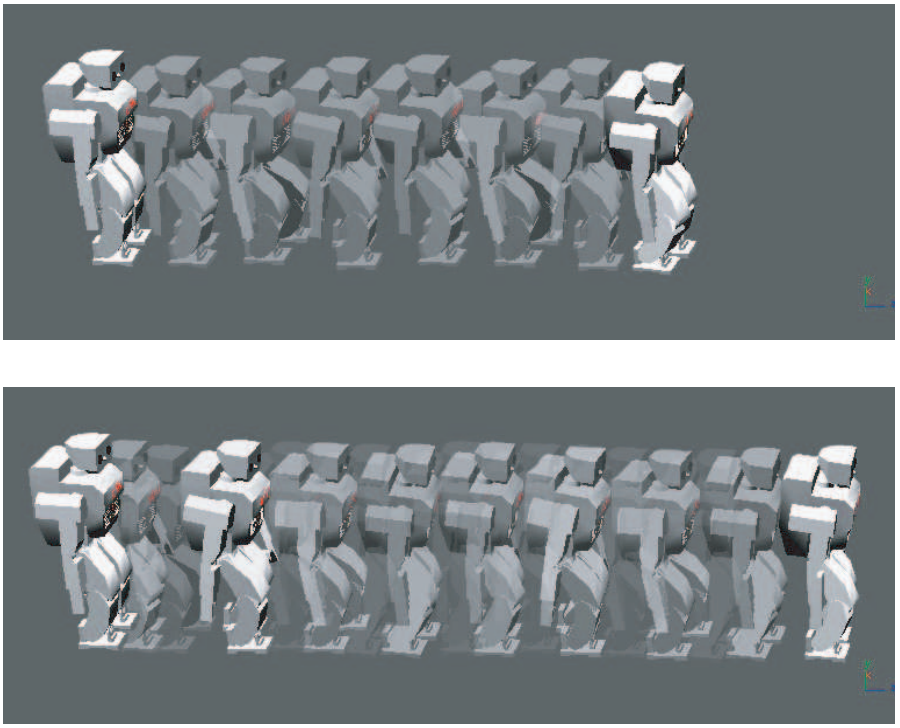

FIGURE 4.7: Initial and optimized walking gait comparison. The top image depicts the robot performing the initial walking gait for 2 seconds. Motion-phase optimization results a faster walking gait as shown in the bottom image.

indicates that the robot has learned to walk forward with greater stability. Figure 4.7 shows that the optimized walking gait is significantly faster than the original one. The walking speed of the optimized walking gait were able to increased to three times the original gait in further optimization. Thus, a conclusion is that φ is controlling the timing of the motion.

4.4 Three-dimensional Optimization

The second experiment is focused on three-dimensional optimization of the initial walking gait based on Equations 4.7 to 4.13. Since the optimization process is performed in the three-dimensional of $\boldsymbol{\phi}$, $\mathbf{r}$ and $\mathbf{h}$, in cylindrical coordinate system $\boldsymbol{\Phi}$ novel postures resulting from optimized actions that do not lie on the constraint pattern shall be expected.

The trajectory of the optimized walking gait in the low dimensional subspace of Figure 4.8 was obtained after three episodes of sensory-motor prediction learning. An improved dynamically balanced walking gait is achieved. The new trajectory has a similar shape to the initial one. It has a larger magnitude and is shifted down from initial pattern. After

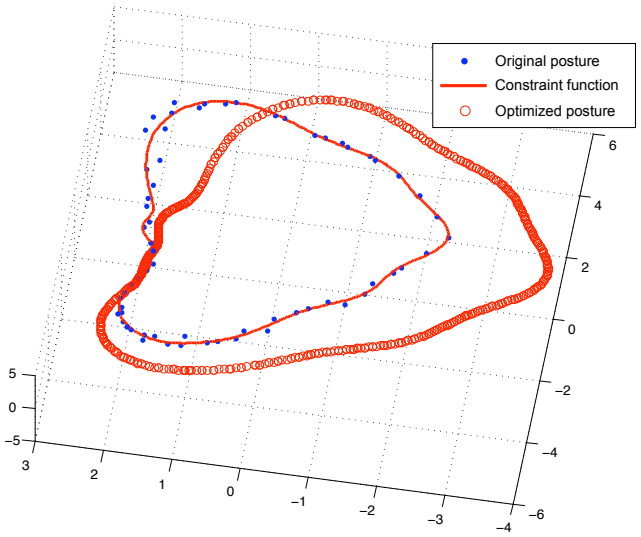

FIGURE 4.8: Three-dimensional optimization results for a walking motion pattern in a low-dimensional subspace based on an action subspace embedding constraint.

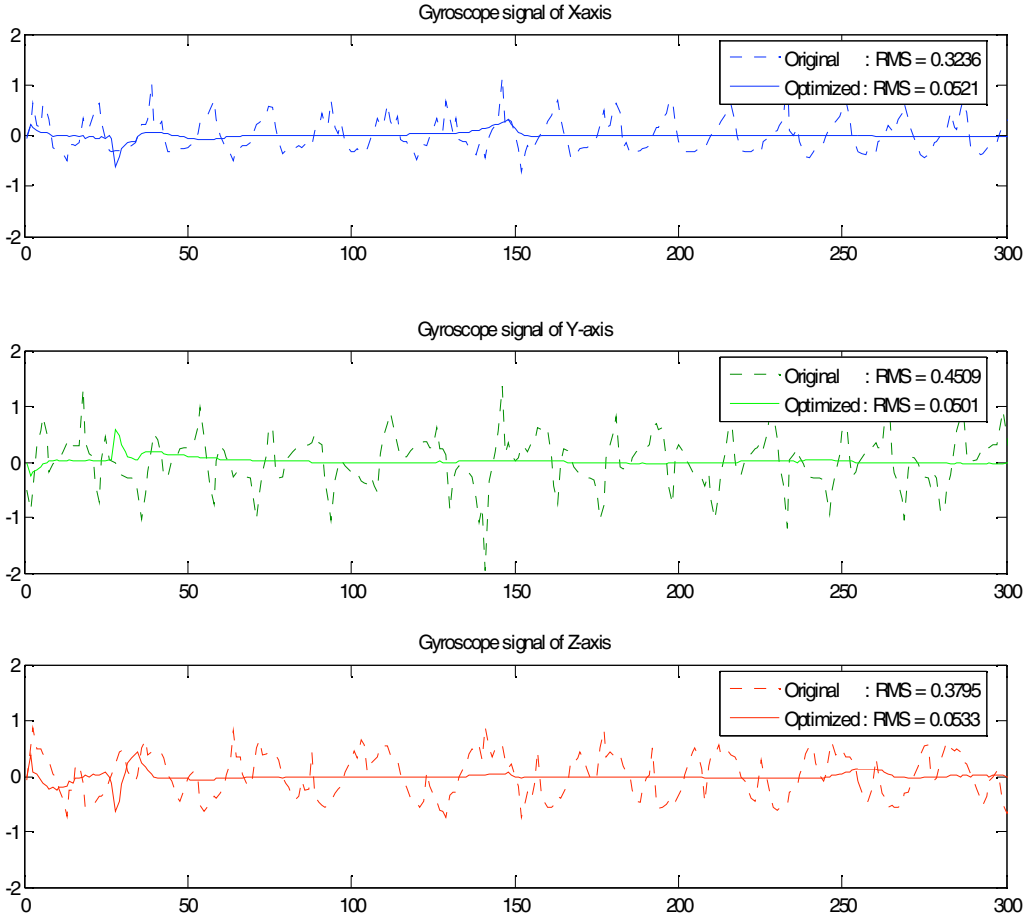

FIGURE 4.9: Comparison of gyroscope signals from initial and optimized walk. The plots from top to bottom show the gyroscope signals for the axes **x**, **y** and **z** recorded during initial and optimized walking motions. Notice that all of the Root mean squared (RMS) values are significantly reduced for the optimized motion.

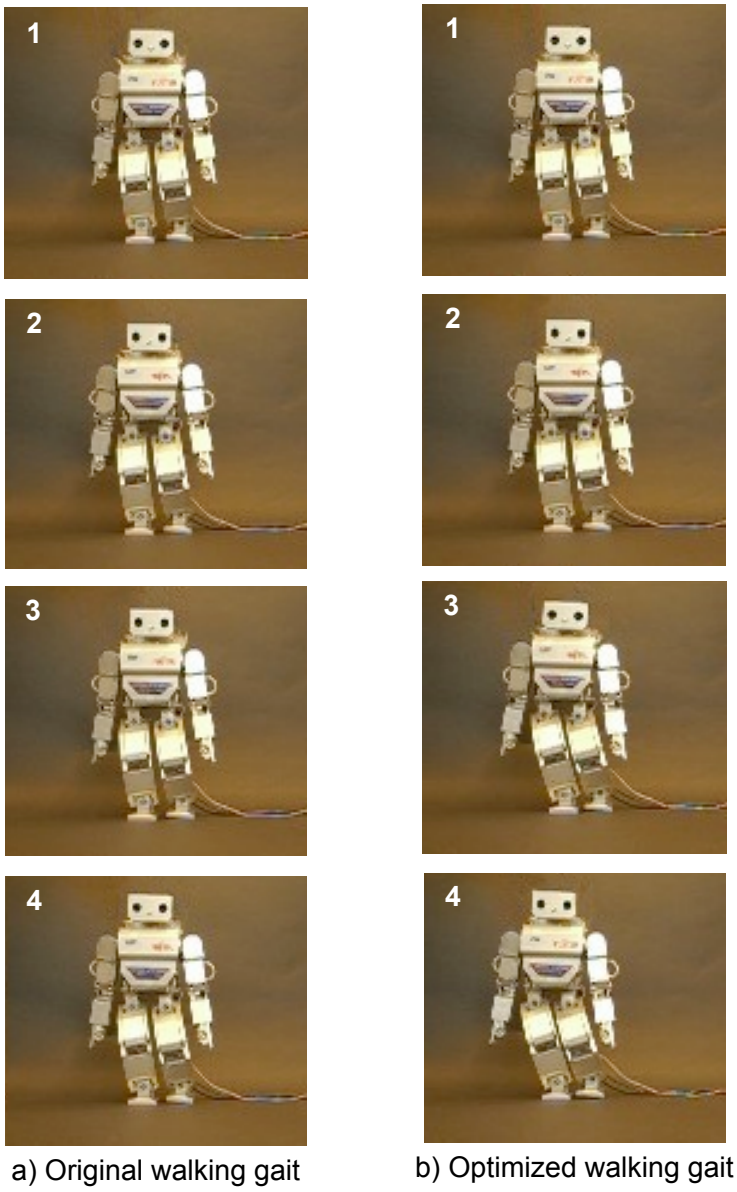

FIGURE 4.10: Three-dimensional optimized walking gait on the Fujitsu HOAP2 robot.

remapping this trajectory back to the high dimensional space, The optimized motion pattern is tested with the simulator and the real robot. The gyroscope reading of the new walking pattern is shown in Figure 4.9. The RMS values of the optimized walking gait along the **x**, **y** and **z** axes are 0.0521, 0.0501 and 0.0533 respectively, whilethe values for the original walking gait were 0.3236, 04509 and 0.3795. The RMS values from the optimized walking gait are significantly less than the original walking gait. This indicates significant improvement in the dynamic stability of the robot. These results are consistent with Equation 4.7. The robot walks with larger step but slower walking speed than the original walking gait. The optimized and original gaits are shown in Figure 4.10. The optimized walking gait has a different balance strategy from the

original walking gait. For the original gait, the robot quickly swings the whole body on the side of the support leg while it moves the swing-leg forward. For the optimized gait, the robot leans on the side of the support leg, bends the torso back in the opposite direction while it moves the swing leg forward slowly. With the optimized gait, the robot also keeps its torso straight up in the vertical direction all the time. Figure 4.9 confirms that the algorithm was able to optimize the motion in such a way that the gyroscope signals of the optimized motion are almost flat.

4.5 Summary

The model-predictive motion generator plans an optimal complete cycle of motion based on sensory feedback prediction of a predictive model. The predictive model is constructed by time-delay RBF network.The predictive model learns to predict future sensory inputs based on history information of states and actions. Then, a motion optimization algorithm searches for an optimal eigenpose action command, which is subjected to an objective function to construct a cycle of motion. One-dimensional optimization or motion-phase optimization is a line-search. The result of the optimization remains on the constraint pattern. No new posture is derived from this one-dimensional optimization. However, the motion phase angle was adjusted in an optimal way of learning based on sensory feedback. A set of novel postures were derived from three-dimensional optimization. Gyroscope signals of the 3-D optimized motion shown how effective the motion optimization is.

Chapter 5

Learning to Walk through Imitation

In previous chapters, the methodology of whole-body optimization of humanoid motion in low dimensional subspaces has been created. Joint angle data recorded from a humanoid robot were transformed into 3-D eigenposes by using PCA. A predictive model of causal relationship between three-dimensional posture commands and their consequent gyroscope signals was obtained through the time-delay RBF network. Then, an optimization algorithm utilized the predictive model to derive a new optimal walking gait. In this chapter, instead of using data that were recorded from a robot, human motion captured data (mocap data) are used, instead. There are two problems that make the mocap data can not be applied directly on a humanoid robot. First, number of joints and joint type between a humanoid robot body and a human body are generally different. This problem is known as *the correspondence problem* in imitation learning literatures. Second, there are high degree of differences of dynamics between the two bodies, the mocap data initially generate dynamically unstable motion on a robot. In this chapter, a heuristic kinematic mapping technique for solving the correspondence problem is explained. The action subspace scaling in section 3.4 is employed to initialize and facilitate the learning process. A result of learning a human walking gait via imitation is shown in this chapter.

5.1 Human Motion Capture and Kinematic Mapping

The correspondence problem is a crucial problem in the research area of learning by imitation. It is the problem of searching for the best match of the features of interest. For our particular case here, the problem is kinematic mapping of the whole-body postures

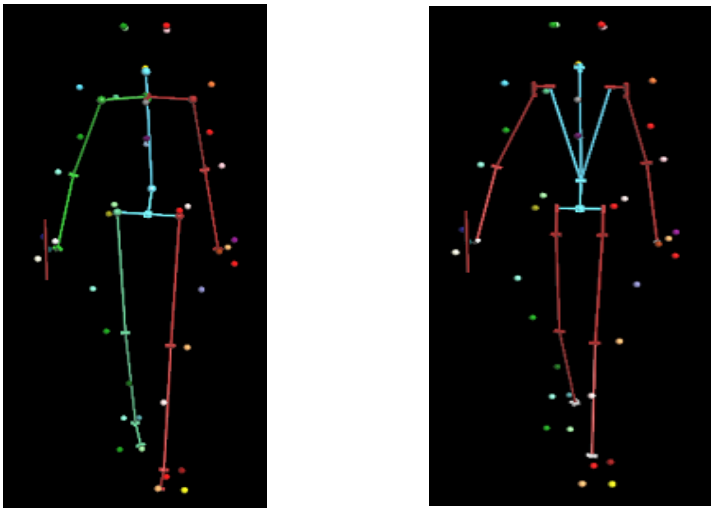

FIGURE 5.1: Human skeleton (left) and robot skeleton (right) for kinematic mapping.

between a human demonstrator and a Fujitsu HOAP-2 humanoid robot. The human subject and the robot share similar humanoid appearances, however their kinematic structure (skeleton) are dissimilar. In this dissertation, the correspondence problem is solved by searching for a set of joint angle data of the robot that generates the best match poses to the human demonstrator. The solution is simply obtained by solving inverse kinematics (IK) of correspondent marker positions on the human body and body of the robot. Initially, a set of markers is attached to the human subject and the 3-D positions of these markers are recorded for each pose during motion. A Vicon optical system running at 120Hz and a set of 41 reflective markers were used. These recorded marker positions provide a set of Cartesian points in the 3D capture volume for each pose. To obtain the robot's poses, the marker positions are then assigned as positional constraints on the robot's skeleton to derive the joint angles using standard IK routines. As depicted in Figure 5.1, in order to generate robot joint angles, the human subjects skeleton is replaced with a robot skeleton of the same dimensions of human skeleton. For example, the shoulders were replaced with three distinct 1-dimensional rotating joints rather than one 3-dimensional human ball joint. The IK routine then directly generates the desired joint angles on the robot skeleton for each pose. There is a limitation of this technique. There may be motions which the robots joints cannot approximate the human pose in a reasonable way. This means we should only demonstrate action that the target robot can perform. For example, using toes in the demonstrated walking gait is avoided. In the case of the arms movement, the target robot is HOAP-2 robot, which has only four degree of freedoms at each arm, demonstration of actions that require six degree of freedoms is also avoided. However, only in classes of human motion that the robot can handle are considered, this method proved to be a very efficient way of

generating large sets of human motion data for robotic imitation. Figure 5.2 shown 3-D eigenposes with some examples of the corresponding human pose of a walking cycle from motion capture data.

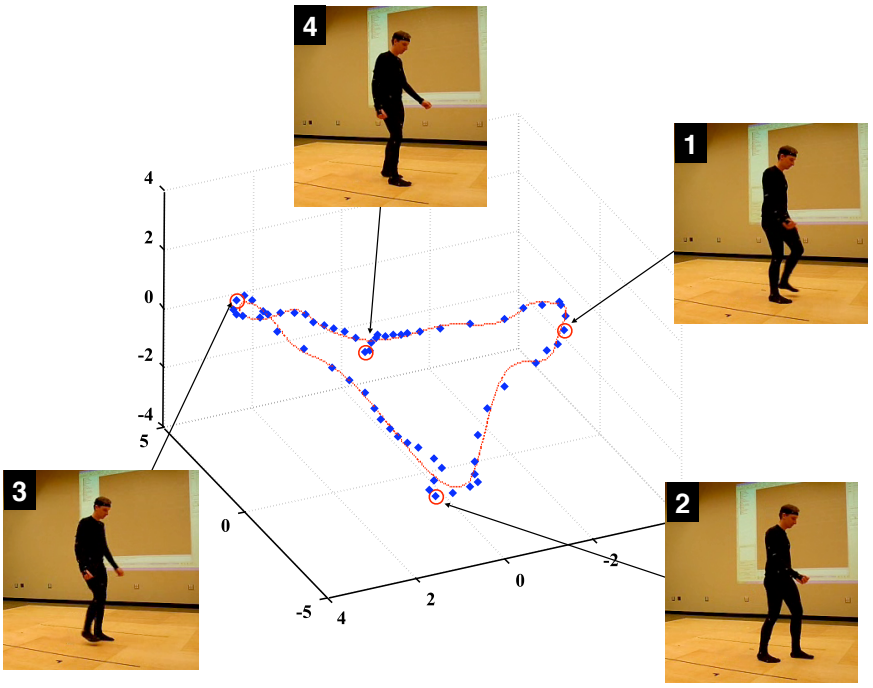

FIGURE 5.2: Posture subspace and example poses from mocap. Linear PCA was applied to joint angle data that is mapped from a human kinematic configuration through motion capture system as described in Section 5.1. Blue diamonds along the function approximated trajectory represent different human postures during a single walking cycle. Red circles mark various example poses as shown in the numbered images.

5.2 Optimization of motion capture data

This experiment focused on making robot learns how to walk using human mocap data. Optimization of the walking pattern from human mocap data is difficult because the initial gait is initially unstable. Thus, a motion scaling strategy in a low-dimensional subspace as described in section 3.4 is employed. When the initial walking pattern in the low-dimensional subspace is scaled down, it produces a smaller movement of the humanoid robot, resulting in smaller changes in dynamics during motion. The initial pattern is scaled down until a dynamically stable motion is found and the learning process is started. The motion optimization method in Section 4.2 is applied to the scaled-down pattern until its dynamic performance reaches an optimal level; the trajectory of the optimization result is gradually scaled up toward the target motion pattern. In this experiment, the scaling of 0.3 of the original motion pattern is typically stable enough to start the learning process. The final optimization result obtained using this

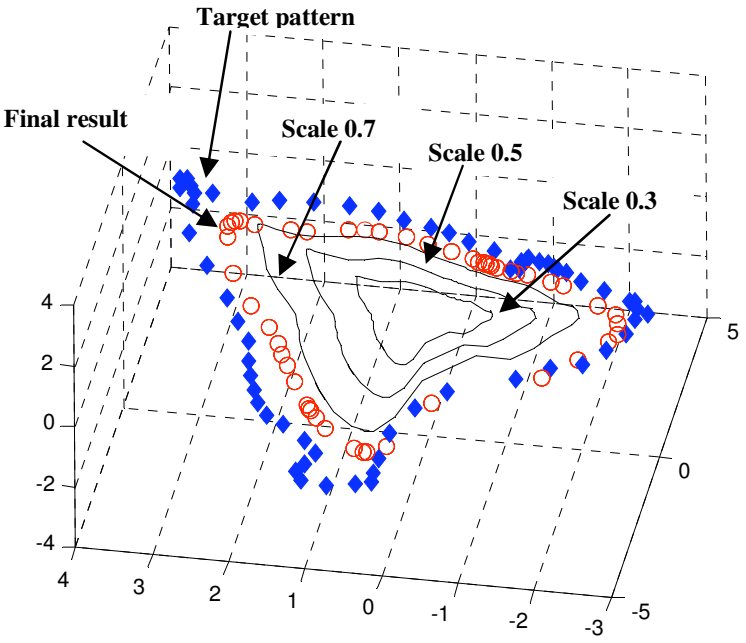

FIGURE 5.3: Motion pattern scaling and optimization of human mocap data. The target motion pattern is scaled down until it can produce a stable motion to start the motion optimization process.

procedure is shown as a trajectory of red circles in Figure 5.3. It corresponds to about 80 % of the full scale motion from mocap data.

For the results in Figure 5.3, five learning iterations for scale 0.3, 0.5 and 0.7 were performed. And, ten learning iterations were performed for the final results for the scale 0.8. The optimization time also depends on parameters $\epsilon_\varphi, \epsilon_r$ and ϵ_h. The parameter ϵ_φ must be defined such that value of φ_s is greater than the maximum difference of motion-phase-angle of the original mocap data. This will ensure that the optimization algorithm is allowed to search for a pose in a range that the original movement achieved. The longer range of φ_s is the better exploration. For r and h, the same parameter setup with φ could be applied. The value of ϵ_r and ϵ_h were set to 0.5 for all of the optimizations. The objective function in (4.6) has three tuning parameters, which are λ_x, λ_y and λ_z. At the beginning, values of these parameters are usually set to 1. From observation of the first learning iteration, the parameters may be tuned. After that values of parameters are maintained for the rest of learning iterations. In this dissertation, λ_x and λ_z were set to 1.0. While λ_y, which correspondent to the vertical direction was set to 2.0. Because, a lot of unexpected turns during the first learning iteration motion were noticed.

Simulation and experimental results are shown in Figure 5.4. The learning process is performed in the simulator [34] and tested the resulting motion on the real robot. The walking gait on the real robot is not as stable as the results in the simulator because of differences in frictional forces modeled in the simulator and in the floor. However,

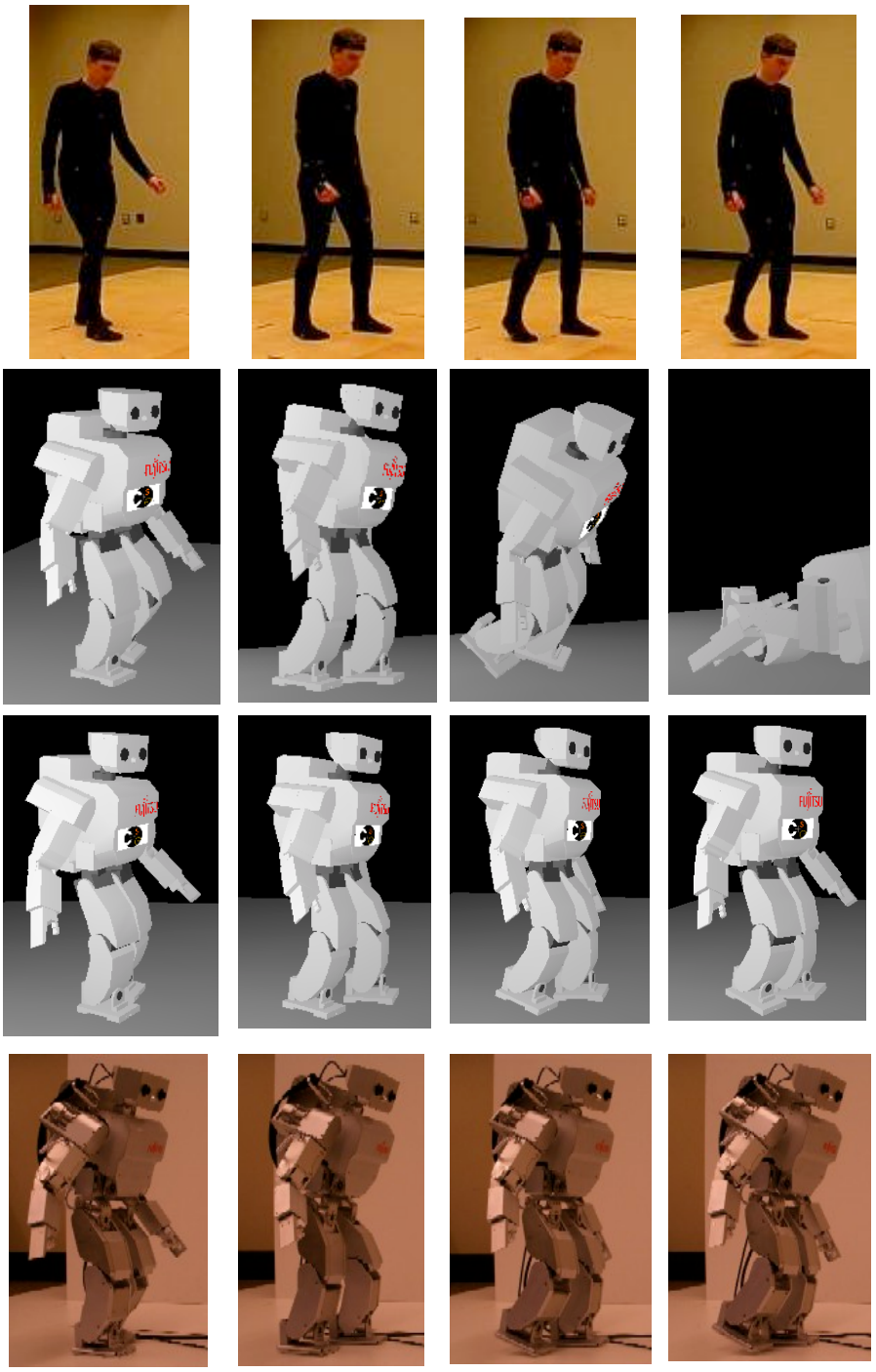

FIGURE 5.4: Learning to walk through imitation. The pictures in the first row show a human subject demonstrating a walking gait in a motion capture system. The second row shows simulation results for this motion before optimization. The third row shows simulation results after optimization. The last row shows results obtained on the real robot.

performing further learning directly on the real robot (if permissible) should rectify this problem and improve performance. Note that the learned motion is indeed dynamic and not quasi-static motion because there are only two postures in our walking gait that can be considered statically stable, namely, the two postures in the walking cycle when the two feet of the robot contact the ground. The rest of the postures of the walking gait do not need static balance condition to maintain balance.

5.3 Summary

Learning of a human walking gait through imitation was demonstrated in this chapter. A heuristic method for solving the correspondence problem was implemented for mapping of human motion data to a robot body. The action subspace scaling technique that diescribed in Chapter 3 was employed to obtain a stable walking gait to start the learning process. After the first small scale motion was learned, the same action subspace scaling technique is used to scale-up the motion pattern toward the original pattern to achieve the original motion as much as possible.

Chapter 6

Motion Optimization in Hyperdimensional Subspaces

Since the beginning of this dissertation, the eigenpose data that have been used for motion learning are three dimensional. As described previously, 3-D data are convenient for visualizing and developing motion optimization algorithm through analytical geometry. Periodic motion patterns such as hand coded walking gait and human mocap walking gait were successfully learned by the algorithm that uses 3-D eigenposes. However for some motion patterns, using only three dimensions of eigenposes can not preserve significant characteristic of the original motion. In this chapter, number of dimensions of the eigenpose data for motion optimization will be extended beyond three-dimensional. The phase-motion optimization concept in section 4.3 will be implemented along with a newly developed cylindrical coordinate transformation technique for hyperdimensional subspaces. The extended algorithm will be used for HOAP-2 robot to learn a sidestep motion from a human demonstrator through a motion capture system.

6.1 Human motion capture data of sidestep motion

A motion capture session of a human demonstrator performing a sidestep motion that is used as the target imitated motion is shown in Figure 6.1. In the figure a human demonstrator was sidestepping to the right hand side of himself. The motion sequence can be divided into four major steps starting from a standing posture. First, the right leg swings off. Second, the right leg lands to the ground. Third, the left leg takes off the ground. And fourth, the left leg swings in toward the right leg. For purposes or later discussion, the sidestep motion shall be defined in four phases, which are *swing-off, landing, take-off* and *swing-in*. After the kinematic mapping process in section

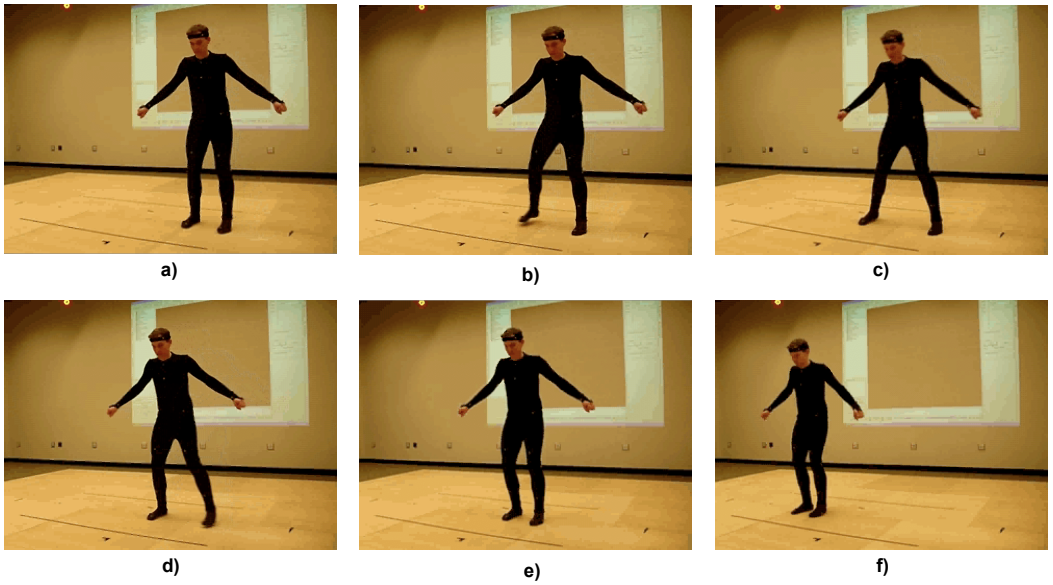

FIGURE 6.1: Motion capture session of sidestep motion. Six samples of a right-hand side sidestep motion sequence are shown in subfigure a), b), c) to f), respectively. A human demonstrator was sidestepping to the right hand side of himself. The motion sequence can be divided into four major steps starting from a standing posture in subfigure a). First, the right leg swings off in subfigure b). Second, the right leg lands to the ground in subfigure c). Third, the left leg takes off the ground in subfigure d). Fourth, the left leg swings in toward the right leg in subfigure e). And come back to a standing posture in subfigure f). Note that the subfigure f) is a standing posture after one more sidestep motion cycle was performed. And each sidestepping cycle takes about 1 second. The sidestep motion shall be defined in four phases, which are *swing-off, landing, take-off* and *swing-in*.

5.1 was applied to the mocap data, 20 dimensions of joint angle data were obtained. Subsequently, the joint angle data were transformed into orthogonal principal axes as in PCA mapping in section 3.1.

The accuracy accumulation along number of principal components are plotted in Figure 6.2. From this figure, when the first three principal axes are used, reverse PCA mapping can recover only 81.38% of accuracy of the original joint angle data. More than 98% of accuracy can be recovered when more than 10 dimensional eigenposes are used. And, 100% of accuracy can be obtained only when all of 20-dimensional eigenpose data are used. In this chapter, 20-dimensional eigenposes are used for learning.

Three-dimensional eigenpose data of the first three principal components of the sidestep motion are plotted in Figure 6.3 as black diamond markers. The pattern of blue dot markers in Figure 6.3 is the sidestep motion pattern when a scaling factor 0.5 (described in section 3.4) was applied, which found to be stable enough to begin the learning process. The phase-motion optimization will be performed on this 0.5 scaled pattern of the full scale motion.

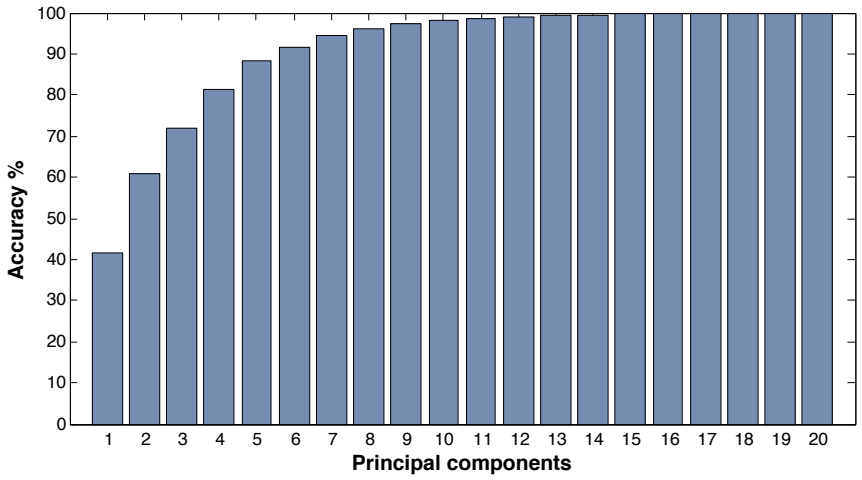

FIGURE 6.2: Accuracy accumulation along principal components of sidestep motion data in figure 6.1. When the first three principal axes are used, reverse PCA mapping can recover only 81.38% of accuracy of the original joint angle data. In order to recover more than 98% of accuracy, more than 10 principal axes have to be used. And, 100% of accuracy can be obtained only when all of 20 principal axes are used.

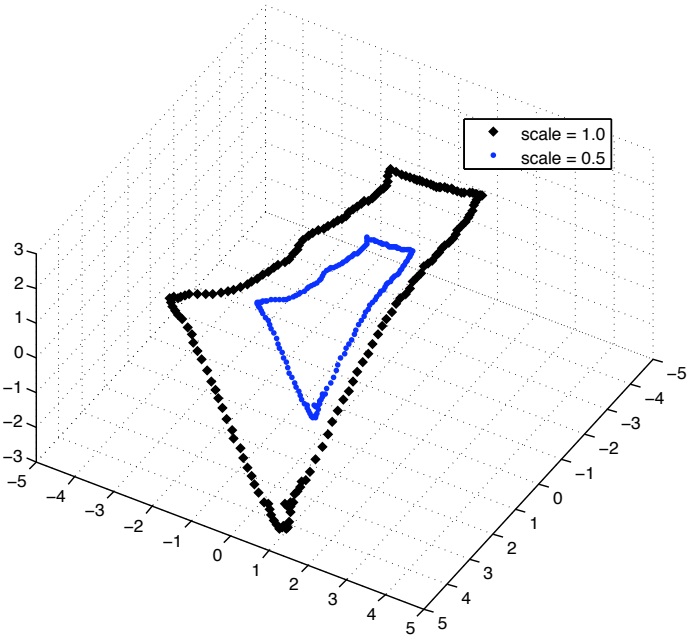

FIGURE 6.3: First three dimensions of sidestep eigenpose data at scale 1.0 and 0.5.

6.2 Cylindrical coordinate transformation of hyperdimensional subspaces

The motion-phase optimization must be performed in a cylindrical coordinate system. Transformation of data from a 3-D Cartesian coordinate system to a 3-D cylindrical coordinate is straightforward. However, that is not the case for transformation of data that have more than three dimensions. In this section, a newly developed concept of cylindrical coordinate transformation for a hyperdimensional data is introduced.

For $f \in \mathbb{R}^n$ when $n = 3$, transformation from a Cartesian space $\mathbb{X}$ to a cylindrical coordinate system $\mathbf{\Phi}$ is mapping:

$$f(x, y, z) \rightarrow f(\varphi, r, h) \tag{6.1}$$

where

$$\varphi = \arctan(\frac{y}{x}), \tag{6.2}$$

$$r = \sqrt{x^2 + y^2}, \tag{6.3}$$

and

$$h = z. \tag{6.4}$$

For $f \in \mathbb{R}^n$ when $n > 3$, a function of n-dimension may be written by

$$f(d_1, d_2, d_3, \ldots, d_n) \tag{6.5}$$

where d_i when $i = 1, \ldots, n$ and $n > 3$, represents a variable in an orthogonal axis in $\mathbb{R}^n$.

We can also express function (6.5) in a form of:

$$f(x, y, z_1, \ldots, z_{n-2}) \tag{6.6}$$

where z_i when $i = 1, \ldots, n-2$ and $n > 3$.

Since, the cylindrical coordinate system is a 3-dimensional coordinate system. Transformation of a hyperdimensional function $f \in \mathbb{R}^n$ where $n > 3$ to cylindrical coordinate system $\mathbf{\Phi}$ is undefined. However, the hyperdimensional function f can be represented by a set of multiple cylindrical coordinate frames. Suppose that f is a 5-dimensional function, f can be express in the form of Equation 6.6 by:

$$f(x, y, z_1, z_2, z_3). \tag{6.7}$$

Mapping of f in (6.7) can be regarded as:

$$\begin{array}{lcl} f(x,y,z_1) & & f(\varphi,r,h_1) \\ f(x,y,z_2) & \Rightarrow & f(\varphi,r,h_2) \\ f(x,y,z_3) & & f(\varphi,r,h_3) \end{array} \tag{6.8}$$

where transformation of φ and r follow equation (6.2) and (6.3). And, the transformation of h_1, h_2 and h_3 is shown in equation (6.4). Thus, mapping of a n-orthogonal dimensions of f to multiple cylindrical coordinate systems can be defined as:

$$f(x,y,z_1,\ldots,z_{n-2}) \rightarrow f(\varphi,r,h_1,\ldots,h_{n-2}). \tag{6.9}$$

For the 20-dimensional of sidestep eigenpose data, 18 cylindrical coordinate frames are needed to describe the data-set. Example of the first six cylindrical coordinate frames of the sidestep motion are shown in Figure 6.4. Let $f_{sidestep}(\varphi,r,h_1,h_2,\ldots,h_{18})$ be a function in the cylindrical coordinate systems that describes the sidestep motion. Subfigure a), b), ..., f) depicts $f_{sidestep}(\varphi,r,h_1), f_{sidestep}(\varphi,r,h_2),\ldots,f_{sidestep}(\varphi,r,h_6)$, respectively. Notice that both subfigure a) in Figure 6.4 and the bule dot makers in Figure 6.3 represent $f_{sidestep}(\varphi,r,h_1)$ but at different perspective. An opened gap can be observed in the subfigure a) as well as in the subfigure c) and f). This is because the data pattern was manually segmented from a human motion capture data sequence that contains multiple periods of sidestep motion. And because of a nature of data which were recorded from human movement, each period of motion is not exactly the same throughout the motion sequence. Even the data are not a perfect closed-cruve pattern, its embedded action subspace is modeled as a closed-curve function.

6.3 Motion-phase optimization of hyperdimensional eigenposes

The curse of dimensionality problem will occur, if one try to perform optimization for all of the orthogonal components of the hyperdimensional eigenposes. Thus, the one-dimemsional motion optimization that has been explored in section 4.5 of the 3-D eigenposes shall be extended for the hyperdimensional case in this chapter. For three-dimensional case, the action subspace embedding(described in section 3.3) is a single parameter function of motion-phase angle φ that can derive values of the radius r and the hight h of a periodic motion pattern in a cylindrical coordinate system $\mathbf{\Phi}$. For a hyperdimensional case of n dimensions eigenpose data, the action subspace embedding is a single parameter function of motion-phase angle φ that derives values of

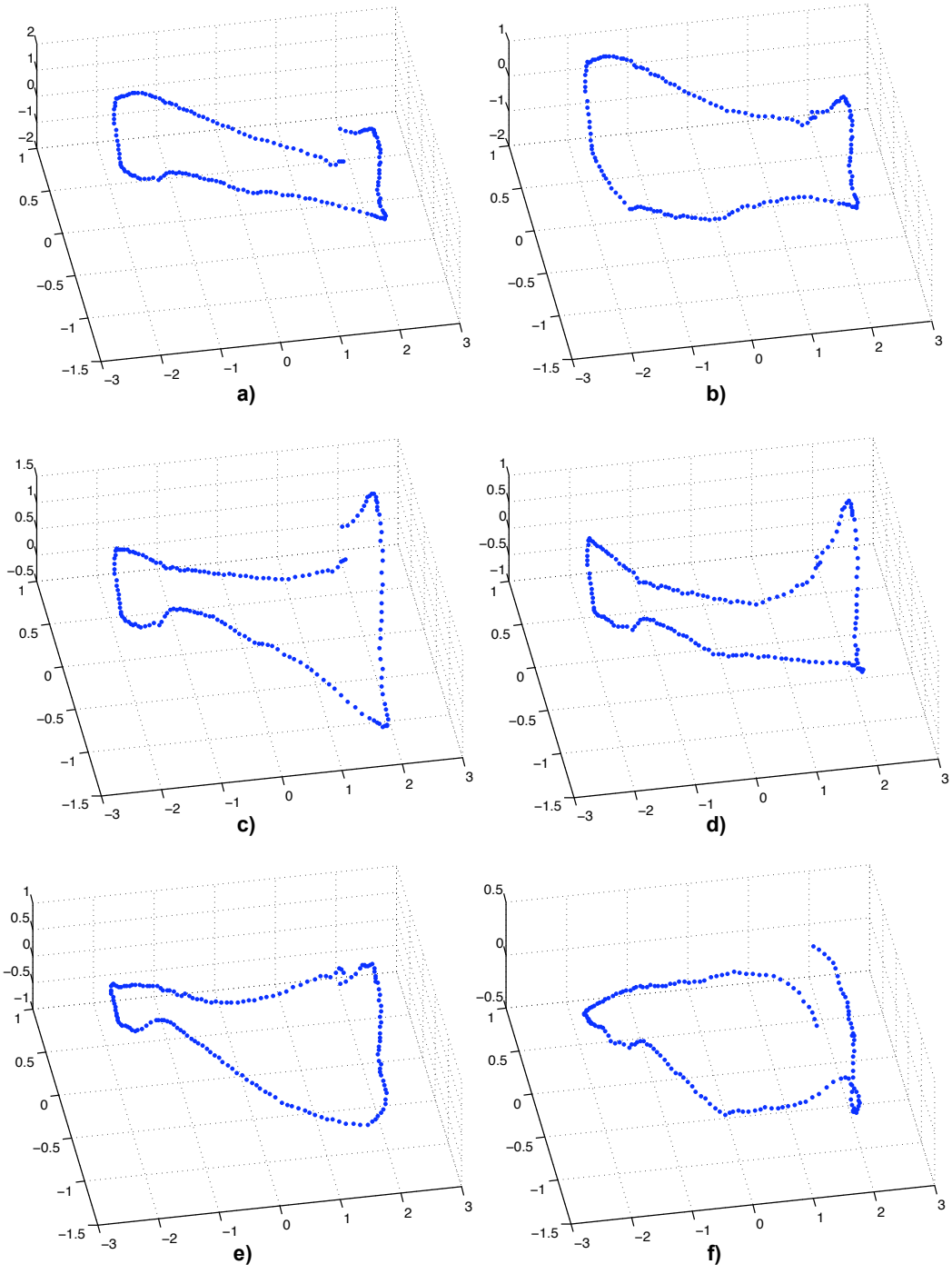

FIGURE 6.4: First six cylindrical coordinates of sidestep hyperdimensional eigenpose data. Let $f_{sidestep}(\varphi, r, h_1, h_2, \ldots, h_{18})$ be a function in the cylindrical coordinate systems that describes the sidestep motion. Subfigure a), b), ..., f) depicts $f_{sidestep}(\varphi, r, h_1)$, $f_{sidestep}(\varphi, r, h_2)$, ..., $f_{sidestep}(\varphi, r, h_6)$, respectively.

$r, h_1, h_2, \ldots, h_{n-2}$ of a periodic motion pattern. For the sidestep motion pattern, the action subspace embedding is one-to-nineteen mapping:

$$[r, h_1, h_2, \ldots, h_{18}] = g(\varphi). \tag{6.10}$$

Note that the cubic spline algorithm is used for modeling the action subspace embedding function in this case instead of the RBF network. The nonlinear autoregressive network with exogenous inputs (NARX) [35] [36] is used for prediction of gyroscope signal instead of the time-delay RBF network. The NARX network is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling. In this dissertation, the model of NARX for predicting gyroscope signal from motion-phase angle input is:

$$\omega_{t+1} = f(\omega_t, \varphi_t, \omega_{t-1}, \varphi_{t-1}). \tag{6.11}$$

Block diagram of the NARX predictor is shown in Figure 6.5. The details of feed-forward neural network structure in Figure 6.5 is shown in Figure 6.6.

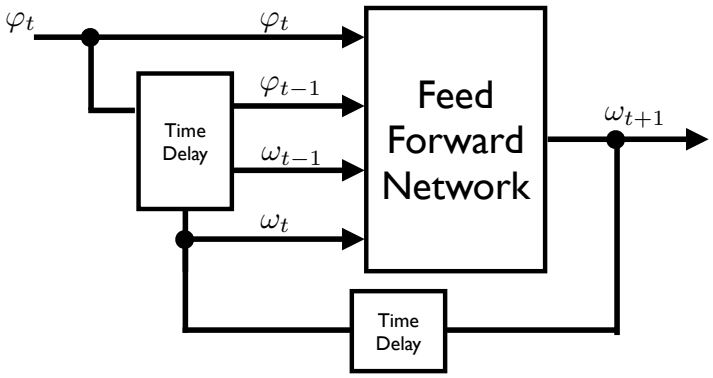

FIGURE 6.5: NARX predictor for motion-phase optimization.

The embedded action in Equation 6.10 and the NARX predictor model in Equation 6.11 can be directly applied to the motion-phase optimization in Equation 4.14:

$$\varphi_t^* = \arg\min_{\varphi_t} \Gamma(F(\omega_t, \omega_{t-1}, \varphi_t, \varphi_{t-1})).$$

Optimization result after five learning episodes is shown in a 3-D coordinate frame of the first three principal axes in Figure 6.7. From this figure, the optimized eigenposes are points on the original motion pattern, but the optimized postures are distributed differently from the original pattern. This is because, the motion-phase optimization is

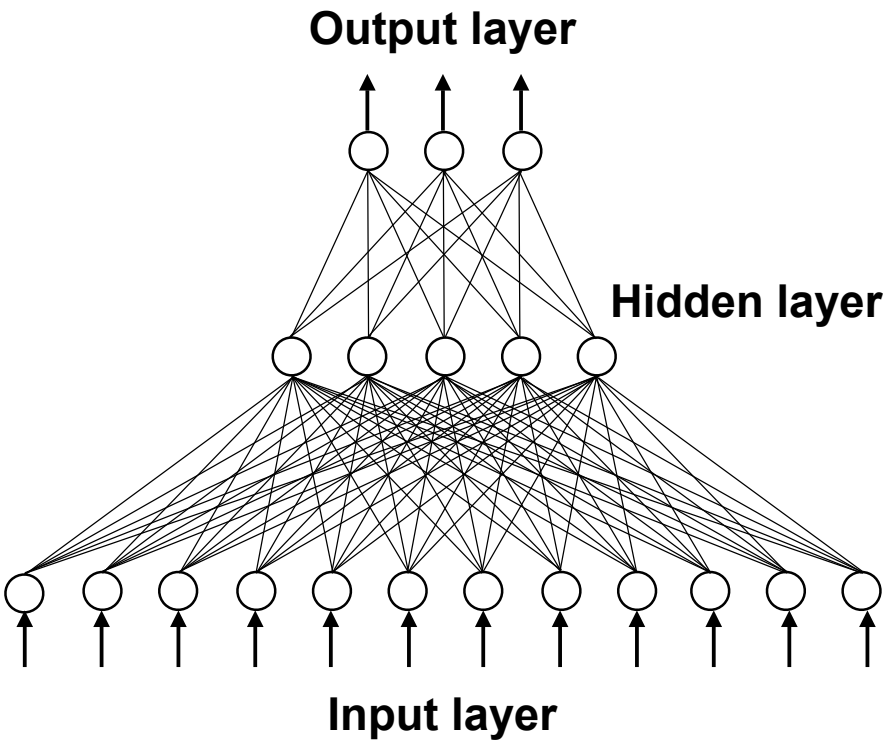

FIGURE 6.6: Feed-forward neural network for NARX predictor. Numbers of node at input layer, hidden layer and output layer are 12, 5 and 3, respectively.

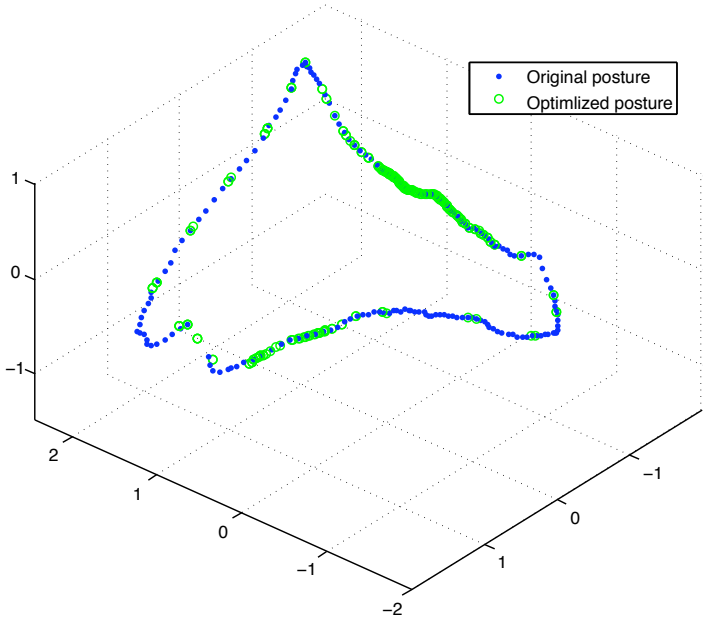

FIGURE 6.7: Phase-motion angle optimization result of sidestep hyperdimensional eigenpose data.

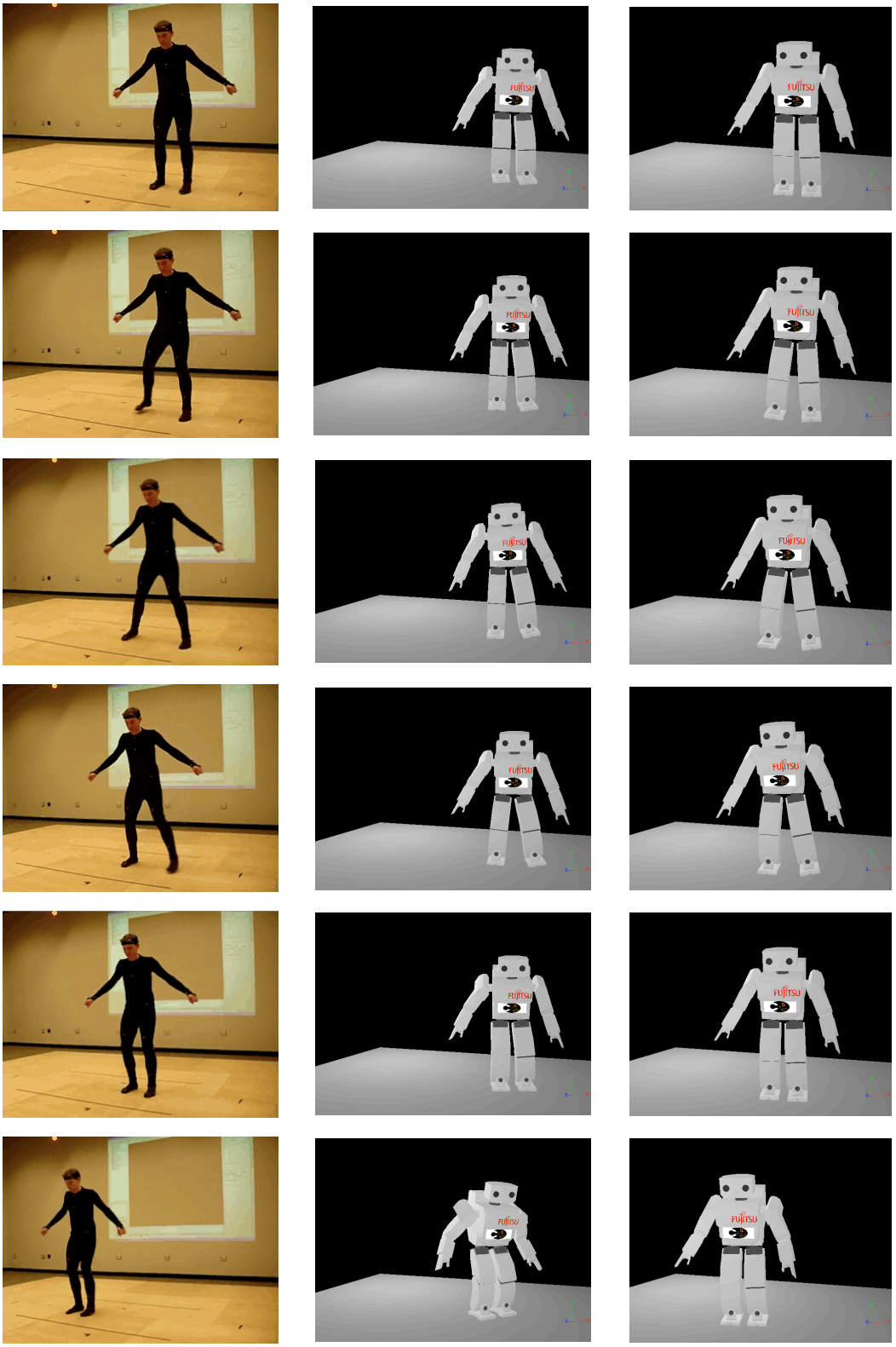

FIGURE 6.8: Simulation result of sidestep hyperdimensional eigenpose data optimization. Subfigures in column a) shown, original sidestep motion sequence of the human demonstrator. Subfigures in column b) shown, the sidestep motion sequence on HOAP-2 robot in a dynamics simulator without optimization at the motion scale 0.5. Subfigures in column c) shown, the sidestep motion after five learning episodes at motion scale 0.5.

one-dimensional optimization on the parameter φ of Equation 6.10. Then the optimized eigenposes are strictly constrained within the original set of postures. Difference of data distribution between the original pattern and the optimized pattern, means timing of postures during the motion have been altered. Notice in the figure that the opened gap of the original pattern is now closed by the optimized postures. There are of two reasons for this phenomena. One is that, the action subspace embedding or the constraint pattern is modeled as a closed-curve. The other is that, based on sensory feedback during learning episodes, the optimization algorithm found that it can achieve a lower gyroscope signal oscillation. Thus, the algorithm assigned some postures in the opened gap. As a result, the movement during the previous opened gap is smoother. Another attempt of the algorithm to obtain smoother movement can be noticed from the most lower-left conner of the pattern in the figure that the algorithm decided to plan the trajectory across an irregular conner of the original pattern.

Simulation results of sidestep motion are shown in Figure 6.8. Column a) in Figure 6.8 shows original sidestep motion sequence of a human demonstrator. Column b) shows HOAP-2 robot performing the sidestep motion sequence at motion scale 0.5 without optimization in a dynamics simulator. Column c) shows the sidestep motion after five learning episodes at motion scale 0.5. In Figure 6.8, the first row and the last row of subfigures in every column are the standing postures at the beginning and the end of the motion sequence, respectively. The second row is the swing-off phase. The third row is the landing phase. The fourth row is the take-off phase. And the fifth row is the swing-in phase. At column b), right foot and left foot of the robot were bouncing at the landing phase and the take-off phase, which caused the robot could not lift its left foot up in the subsequent take-off phase. As a result, the robot dragged its left foot along the ground during the swing-in period. This made the whole body of the robot turned as can be observed from the last two rows of column b). At column c), the robot could perform the sidestep motion without unexpected turn of the body. While, all of the key postures in the figures of column c) look very similar to the postures in column a), timing of movements are significantly different. The landing phase and the swing-in phase of the optimized motion in column c) are relatively slower than the original human motion. These can also be observed in Figure 6.7. In Figure 6.7, there are two parts of the motion pattern that there are high density of the optimized postures distribution. These are corespondent with the slow landing phase and swing-in phase. The slow landing phase and swing-in phase also prevented the robot from dragging its left foot on the ground. As a result, the unexpected turned was not presence. And the robot learned the sidestep motion successfully.

6.4 Summary

In this chapter, conventional 3-D transformation from Cartesian space to cylindrical coordinate system has been extended to be able to work beyond 3-D. The extended transformation can map an n-orthogonal-axes coordinate system into multiple cylindrical coordinate frames. The one-dimensional motion-phase optimization was demonstrated to work along with the hyperdimensional action subspace embedding. This yields a fast learning and very accurate motion imitation.

Chapter 7

Discussion and Conclusion

In this dissertation, a framework for a humanoid robot for learning periodic-type of human motion through imitation has been developed. Human motion data from motion capture system are mapped to the robot body. Motion data in joint space of the robot are transformed into orthogonal axes of principal components using linear PCA. The motion data that were obtained by PCA are called *eigenpose data*. For periodic-type of motion, the eigenpose data are transformed again into a cylindrical coordinate frame or multiple cylindrical coordinate frames depends on whether 3-D eigenposes or beyond 3-D eigenposes will be used for imitation learning. The eigenpose data in the cylindrical coordinate system are then modeled as a single parameter closed-curve function called action subspace embedding. The single parameter of the action subspace embedding φ is defined as *motion-phase angle*. The motion-phase angle is the parameter that has direct effect to timing of movement during the motion. Motion learning in this dissertation is an optimization of sensory feedback from executing the eigenpose motion commands. At the beginning of the learning process, sensory feedback might be recorded from motion that is reproduced via the eigenpose commands from the mocap data. Generally, the reproduced motion is not stable enough to begin the learning process. In order to be able to get a complete cycle of stable motion to begin the learning, the unstable motion must be scaled down. The motion will be scaled down until a complete cycle of stable motion is found. After sensory feedback from the first motion trial is obtained, a predictive model can be formed. The predictive model is trained to predict sensory feedback of the next time step based on history information of the eigenpose command and the sensory feedback itself. Once the sensory feedback of the next time step can be predicted, the eigenpose commands for a complete motion cycle is planed such that for each action command executed its consequence feedback is optimal subject to the objective of the motion. An optimal action plan can only be obtained when sensory prediction is accurate. The predictive model can deliver more accurate prediction only

when it has more history information of action and feedback. Thus, through a number of learning trails an action plan for a good complete motion cycle will be achieved. This is a brief summary of the overall motion imitation learning framework. Subsequently, each individual part of the framework will be discussed in details based on all of the results in this dissertation.

7.1 PCA for dimensionality reduction of motion data

Dimensionality reduction techniques are normally used for visualization of high-dimensional data. There very few to none of work that further uses reduced-dimensional data. Because, there is a presumption that essential properties of the data may not be preserved after dimensionality reduction. The work in this dissertation demonstrated merit of the low-dimensional data. The compact representation of whole-body posture in low-dimensional subspaces is the first key component the framework. It allows a tractable motion optimization. The conventional linear principal components analysis is used in the framework for dimensionality reduction of the high dimensional joint data. Linear PCA was first chosen at the beginning of this study for a number of reasons. First, it is a nonparametric algorithm. Its outcome is very reliable. Second, its computation operation is relatively low. So, it is a fast algorithm. Third, it is a well-known algorithm. Thus, it is suitable for serving with a novel idea such as the work in this dissertation.

From results of studies of 3-D posture data from PCA transformation or *eigenposes* can be concluded that temporal sequence of postures during motion is preserved in the PCA low-dimensional subspaces. Manipulations of eigenpose data such as generalization, scaling and sensory feedback mapping can be used for improving dynamic stability of the motion. It will be very interesting to extend this result for different kind of data and in different application other than human motion learning. Manipulation of 3-D eigenpose data such as translation and rotation of the motion pattern have not been in-deep studied. Without a cylindrical coordinate system, eigenpose data of each principal component can be plotted as a function of time. Periodic eigenpose data will be shown as periodic signal with time as in Figure 3.3. From some preliminary results [1], translation of the eigenpose data along an principal component axis is shifting mean value of a principal axis, which could lead to posture shifting. While rotation of the 3-D eigenpose pattern, is a complicated case of magnitude projection between orthogonal axes of the principal components. These hypothesises of translation and rotation of 3-D eigenpose data should be further study in further works.

[1] Not present in this dissertation

From Figure 3.4 and 6.2, accuracy of data that PCA can preserve increases in order with the number of principal components. The motion optimization algorithm in this dissertation used only three principal components in Chapter 4 and 5. However, Figure 6.2 in Chapter 6 shown that using only 3-D eigenpose data may not sufficient originality of the prototype motion to learn the motion. Thus, the motion optimization algorithm was extended to work with eigenpose data that have number of dimensions beyond three by using the rest of the principal components. This demonstrated flexibility of linear PCA algorithm. Nonlinear PCA (NLPCA) algorithm, which provides better accuracy data from dimensionality reduction does not have the same flexibility. The nonlinear principal components are not arranged in order. And in the case that more number of nonlinear principal components are required, the auto-associative NLPCA network has to be retrained. Moderns dimensionality reduction algorithms such as locally linear embedding (LLE) [27] and ISOMAP [28] provides choices of number of dimensions of the feature data. However, these two algorithms are not nonparametric algorithms. The results are sensitive to some their tuning parameters. Adding new data to the already learned data set may also involve relearn all of the data set. For linear PCA, the an existed transformation matrix can be used with new data immediately. The Gaussian process latent variable method for dimensionality reduction [32] provides robustness against missing data issue. However, its latent space is not continuous. This could fail the action subspace scaling. Furthermore, an assumption of linear combination of principal components of linear PCA also supports optimization of individual dimension of eigenposes. This will prevent the number of computational operations of exhaustive search (optimization), which is used in this framework to grow exponentially. Thus, linear PCA is the most suitable dimensionality reduction algorithm for human motion learning application.

7.2 CNLPCA for periodic motion recognition

In the preliminary study of this dissertation, CNLPCA is proposed for motion recognition of a humanoid robot. In a space of reduced dimensionality, the algorithm is able to divide sequences of humanoid motion data into segments of periodic motion. The motion recognition algorithm has two phases. The first phase is a temporal ordering segmentation process that combines learning and temporally-constrained data point assignment among multiple neural networks. The second phase is a process of minimizing redundant networks that merges redundant networks based on the average spatial distance between the periodic trajectories in the feature space.

CNLPCA performs well for periodic humanoid motion patterns. Note that, although the joint angle space used in this research is 20 dimensional, the proposed algorithm can deal with more than 20 dimensions. The algorithm abstracted five out of six types of humanoid motion without any prior information about the number or type of motion patterns. There are three tuning parameters: n, α, and the threshold for D_{ij} in the algorithm. While values for these parameters cannot be randomly assigned, the automatic segmentation results are not sensitive to them. One may refer to the guideline for assigning values to these parameters in section 2.2.

Although a CNLPCA neural network divides and conquers the low-dimensional data in the feature space along the temporal sequence, it cannot distinguish motion patterns that only differ in frequency, because a CNLPCA network is a static network. In other words, our algorithm cannot recognize the differences between fast and slow motion patterns that are otherwise kinematically identical, if such patterns exist. However in practice, a fast walking gait and a slow walking gait have different postures because of the change in dynamics. This produces different low-dimensional data patterns in the feature space. Thus, the algorithm will be able to distinguish these data patterns. The automatic segmentation results can be used to facilitate the learning of complex tasks performed by humans by deriving an abstract state-action space for reinforcement learning [24].

The fundamental concept of CNLPCA that one variable can govern cycle of periodic motion is an interesting idea. In this dissertation, this concept was later used for construction of action subspace embedding. In the preliminary study, training the NLPCA network takes very long time for a large number of data. Results of NLPCA are also not reliable. Different results can be occurred from different NLPCA networks that were trained from the same data-set. Performance of CNLPCA is good when it is used for modeling planar circular patterns. But, when a closed-curve pattern appears to be an irregular shape, the CNLPCA can not perform well. Many interesting and useful concepts of low-dimensional human posture have been arisen from the preliminary study in Chapter 2. Consequently, algorithms that have similar properties without the drawbacks that mentioned here were developed in later chapters of this dissertation.

7.3 Cylindrical coordinate system for periodic motion

The eigenpose data pattern of periodic motion always appears as a closed-curve. The NLPCA with circular constraint (CNLPCA) is able to model and generalize a closed-curve pattern. However generalization of CNLPCA can be suffered, when variation of data on the vertical axis is high. This is due to the nature of the underlying single

parameter at its feature layer. A 3-D closed-curve in Cartesian coordinate system can be considered as an opened-curve function in polar coordinate system. Thus, various function approximation techniques can be applied. Cylindrical coordinate system and spherical coordinate system are both polar coordinate system. However, the cylindrical one are more relatively intuitive for transformation back and forth to the cartesian system. In Chapter 6, a concept for transformation of an n-orthogonal axes coordinate system when $n > 3$ to multiple cylindrical coordinate frames was introduced. This allows eigenpose data that have number of dimensions more than three dimensions to be used for the motion learning framework in this dissertation. The hyperdimensional coordinate transformation in section 6.2 can not be done, If the spherical coordinate system was used instead of the cylindrical coordinate system.

7.4 Action subspace embedding

In section 3.3, action subspace embedding models a motion pattern (3.23) as a single angular parameter function in cylindrical coordinate system. It is designed to imitate a characteristic of CNLPCA that can generate a complete periodic function by varying the angular parameter that constrains its feature layer from 0 to 2π. While function approximation of CNLPCA is auto-associative neural network, for the action subspace embedding any function approximator can be used. For a three-dimensional case, the action subspace embedding is constructed to map φ to r and h. The single angular parameter φ is defined as *motion-phase angle* of a periodic motion pattern. The motion-phase angle has direct effect on timing of posture during the motion. This property is used for one-dimensional motion-phase optimization in section 4.3 and 6.3. Unlike the case of three-dimensional optimization (section 4.4 and 5.2), the search space of the motion-phase optimization is only one-dimensional. Its number of calculations does not grow exponentially when the size of search space is increased. Because of the rest of motion parameters in other dimensions are constrained to the motion-phase angle, adjusting the motion-phase angle effects overall movement of the motion. Thus, the motion-phase optimization is fast and effective. However, for learning motion that is highly dynamic, motion scaling and 3-D optimization is unavoidable. For 3-D optimization, the action subspace embedding is used for constructing the 3-D search space as described in section 4.2.

7.5 Action subspace scaling

The action subspace scaling is performed in eigenpose space. This property is not occurred from PCA transformation process. The scaling ability is inherited from a standard statistical data normalization process that is performed as the data preprocessing for PCA. Note that multiplying a vector of raw joint angles data by a scalar factor does not yield a similar posture. Because, the data among each joint originally are in different scales of values. But after normalization, the data are scaled into the same range. When the normalized data is multiplied by a scalar value, the scaling factor contributes proportionally to ranges of the motion of each joint. This makes similar postures to be produced with different magnitudes. The posture scaling yields reasonable results only when the motion data set contains only one specific type of motion. In a general case, motion scaling might be performed directly by scaling the normalized data without involving a manipulation in the eigenpose space as the action subspace scaling. However for motion learning using eigenposes, optimized postures are derived from the eigenpose space. Thus, the action subspace scaling concepts must be employed.

The action subspace scaling is used for increasing or decreasing size of action subspace embedding. To be able to begin learning from human motion data which initially produces an unstable motion, the action subspace scaling technique in section 3.4 must be employed. The action subspace scaling creates a similar motion pattern with smaller scale of movement. A smaller motion pattern produces less dynamic perturbation. An unstable motion pattern is scaled down until at one point, that the motion is stable enough to begin learning process. Scaling down a motion pattern can be done without much precaution. On the other hand, scaling up the magnitude of motion for motion generalization or parameterization purposes, the joint limits of the robot have to be concerned. Dynamically stable motion should not be expected directly from motion scaling process without optimization.

7.6 The predictive model

A predictive model is used for model-predictive motion planing in this framework. It predicts sensory information of the next time step based on history information of sensory-state and action-command. Learning algorithms for time series prediction were implemented to constructed the predictor.

In Chapter 4 and Chapter 5, the predictive model was obtained by training second-order time-delay RBF network. The second-order predictive function was justified by simply testing prediction by increasing the order of the function from first-order function.

Prediction results of second-order function was found to be satisfied as shown in Figure 4.2. A model that has order higher than two are not tested. Because, the second-order function has already give us good results. Moreover, increasing of the order of the function is increasing number of dimensions of the input vector of the predictor. In this case, the number of dimensions of the input vector is increased by six dimensions when an order of the function are increased. To avoid unnecessary computation complexity, the function are kept to be as simple as possible. However, a first-order function was implemented the work in [37], when center-of-pressure was used as the sensory state instead of gyroscope signals. Thus, an appropriate order of the function also depends on the state variables in the model. Different kinds of state variable can be applied to this framework. The time-delay RBF network that was used in this dissertation has 3,000 kernels, which is the maximum number of kernels that can be implemented in the computer, which was used in experiments. A higher number of RBF kernels may result better prediction. But this will increase computational cost, especially when it is used in our optimization routine, which is a brute-force search.

In Chapter 6, the nonlinear autoregressive network with exogenous inputs was used for training the predictive model. Training time and forward calculation time of the NARX are much shorter than the RBF. The prediction results of NARX algorithm are less accurate than results from RBF. However, the NARX network are much more robust when out-of-range input data are presented. The robustness against out-of-range input is very beneficial for predictive motion generator that is used for motion optimization part in this framework. Because larger search-space can be used in the optimization process. As a result, the overall learning process will be accelerated.

7.7 Motion optimization

The predictive model is combined with an optimization algorithm to form a model predictive motion generator. The predictive motion generator resembles the model predictive control scheme [38]. The predictive motion generator with time-delay RBF predictor currently does not work fast enough to be used for online motion controller. For the simplest one-dimensional optimization (in section 4.3), the algorithm took three minutes for optimization of one walking gait cycle. Thus, motion planing process in this dissertation is considered to be batch planing. For three-dimensional optimization (in section 4.4), one walking cycle took about 20-30 minutes. Because the optimization algorithm that is used in this dissertation is a straightforward brute-force search, the overall optimization time is increased exponentially when number of dimensions is increased. A more sophisticated optimization algorithm is planned to use in the future work.

Ten learning iterations were performed to get the optimization result of the hand-coded walking gait in Figure 4.8. For the results of human walking gait from motion capture data in Figure 5.3, five learning iterations for scale 0.3, 0.5 and 0.7 were performed. And, ten learning iterations were performed for the final results for the scale 0.8. The optimization time also depends on parameters $\epsilon_\varphi, \epsilon_r$ and ϵ_h. The parameter ϵ_φ must be defined such that value of φ_s is greater than the maximum difference of motion-phase-angle of the original mocap data. This will ensure that the optimization algorithm is allowed to search for a pose in a range that the original movement achieved. The longer range of φ_s is the better exploration. For r and h, the same parameter setup with φ could be applied. The value of ϵ_r and ϵ_h were set to 0.5 for all of the optimizations. The objective function in (4.6) has three tuning parameters, which are λ_x, λ_y and λ_z. At the beginning, values of these parameters are usually set to 1. From observation of the first learning iteration, the parameters may be tuned. After that values of parameters are maintained for the rest of learning iterations. In this dissertation, λ_x and λ_z were set to 1.0. While λ_y, which correspondent to the vertical direction was set to 2.0. Because, a lot of unexpected turns during the first learning iteration motion were noticed.

Note that even, all of the motion optimization in this dissertation was initially performed in a simulator then tested with the real robot, the learning can be performed directly on the robot as depicted in Figure 1.1. Computer simulations were used for accelerating the result and safety reasons.

7.8 Further research and development

The human-motion learning through imitation framework described in this dissertation demonstrated how a humanoid robot can learn basic human motion. It is aimed to serve as a tool for learning complex human behavior. One immediate application of this work is replacing the hand-coded motions in Chapter 2 with the imitation human motions. A further implementation could be using the imitation human motions as high-level actions for learning complex behavior by learning algorithm such as reinforcement learning. To learn actions other than walking and sidestepping, the objective function in (4.6) could be modified to accommodate different sensory variables such as foot-contact pressure or ZMP. This research direction are currently being studied. Modern learning algorithm such as nonparametric probabilistic inference and learning [37] is also explored to improve performance of the predictive model and the optimization process. The proposed framework work as an off-line motion generator rather than an on-line feedback control. Thus, it cannot be applied directly to the problem of navigation on non-uniform uneven terrain. To effectively navigate on uneven terrain, a higher degree

of compliance control is needed in the leg and foot actuators. Robustness of motion can be added to an off-line motion generator by using a motion stabilizer [39], which is a combination of simple force/torque and gyroscope-based feedback controllers. Possibility for develops a real-time feedback controller based on learning inverse model of predictor (Equation 4.1) is also currently under study. A method based on concepts in this dissertation for learning none-periodic human motion as well as motion parameteriation using eigenposes are also currently study. The researches based on the results in this dissertation are on-going researches.

7.9 Conclusion

A humanoid robot that learns how to perform bipedal locomotion by imitation through representation of whole-body posture in low-dimensional subspace and hyperdimensional space of eigenpose data is successfully demonstrated. The low-dimensional subspace that is used in this work does not only contain sample of demonstrated postures, but also consist of none demonstrated poses. The none demonstrated poses are constructed by variation of the values of the principal components. The none demonstrated poses can be clearly observed from the optimization result in section 4.4. And, notice that the low-dimensional subspace only contains poses those are relevant to the target imitated motion.

A humanoid robot learn to walk by combining a learned sensory-motor model with imitation of a human gait is a result of this work. This approach does away with the need for detailed, hard-to-obtain, and often fragile physics-based models that previous methods have relied on for stable gait generation in humanoids. The results also demonstrate that the physics of a complex dynamical system can be learned and manipulated in a low-dimensional subspace. Using a low-dimensional subspace greatly reduces computational complexity and facilitates the learning process. Since all of the joints are always constrained to encode postures near the ones to be imitated, the low-dimensional subspace reduces the occurrence of unmeaningful or potentially harmful actions such as self-intersection in the learning process. The action subspace embedding in cylindrical coordinate system not only further reduces the dimensionality and complexity, but also provides meaningful variables in the low-dimensional subspace such as *motion-phase-angle* φ and r. Optimization of the motion-phase-angle was shown to be equivalent to optimizing posture timing during the motion, while the radius of the action subspace embedding r reflects magnitude of the motion, which is contributed by the first two principal components of the motion pattern. However, an absolute magnitude of a posture in a motion pattern is $\|\mathbf{x}\|$, where $\mathbf{x}$ is an n-dimensional eigenpose vector. The parameter

h in a cylindrical coordinate system is equivalent to the i-th principal component of n components of PCA transformation, where $i > 2$ and $i \leqslant n$. In other words, there is no transformation for h. Thus, shifting level of action subspace embedding up or down in **h** direction is changing the mean value of a principal component. It depends what basis movement that the principal component represents. The result of shifting value of h is shifting all posture which that principal component produces.

Appendix A

Principal Components Analysis: Direct and Inverse Mapping

This appendix describes how to perform PCA mapping in this dissertation as well as how to perform inverse PCA mapping.

Computing PCA using the covariance method can be perform by the following steps:

Sort data in a form of column vector: Let $\boldsymbol{\theta}$ is a vector whole-body of joint angle data of a humanoid robot that has m joints:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}. \tag{A.1}$$

The time series of joints angle data of a motion segment $\boldsymbol{\Theta}$ can be described in matrix form:

$$\boldsymbol{\Theta} = \begin{bmatrix} \theta_{11} & \theta_{12} & \dots & \theta_{1n} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{m1} & \theta_{m2} & \dots & \theta_{mn} \end{bmatrix} \tag{A.2}$$

where n is number data at each joint.

Normalized the data: Each row of $\mathbf{\Theta}$ must be normalized such that it has zero mean and unity standard deviation. Let μ_i be average value (mean) of joint (row) i:

$$\mu_i = \frac{1}{n}\sum_{j=1}^{n}\theta_{ij} \tag{A.3}$$

and σ_i be standard deviation of joint i:

$$\sigma_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(\theta_{ij}-\mu_i)^2}. \tag{A.4}$$

Let $\mathbf{Q}$ be a normalized version of $\mathbf{\Theta}$:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1n} \\ q_{21} & q_{22} & \dots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \dots & q_{mn} \end{bmatrix} \tag{A.5}$$

where

$$q_{ij} = \frac{\theta_{ij}-\mu_i}{\sigma_i} \tag{A.6}$$

when $i = 1 \dots m$ joint and $j = 1 \dots n$ number of data.

Calculate the covariance matrix of Q: Let $\mathbf{q_i}$ when $\mathbf{i} = 1 \dots m$ be normalized data of joint $\mathbf{i}$. The normalized data $\mathbf{Q}$ can be rewritten in vector form:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q_1} \\ \mathbf{q_2} \\ \vdots \\ \mathbf{q_m} \end{bmatrix}. \tag{A.7}$$

The normalized data of joint i can be expressed in form of row vector:

$$\mathbf{q_i} = \begin{bmatrix} \mathsf{q}_1 & \mathsf{q}_2 & \dots & \mathsf{q}_n \end{bmatrix}. \tag{A.8}$$

Thus, covariance matrix of $\mathbf{Q}$ is:

$$\begin{aligned} \mathbf{A} &= \mathbf{cov}(\mathbf{Q}) \\ &= \begin{bmatrix} \mathbf{cov}(\mathbf{q_1},\mathbf{q_1}) & \mathbf{cov}(\mathbf{q_1},\mathbf{q_2}) & \dots & \mathbf{cov}(\mathbf{q_1},\mathbf{q_m}) \\ \mathbf{cov}(\mathbf{q_2},\mathbf{q_1}) & \mathbf{cov}(\mathbf{q_2},\mathbf{q_2}) & \dots & \mathbf{cov}(\mathbf{q_1},\mathbf{q_1}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{cov}(\mathbf{q_m},\mathbf{q_1}) & \mathbf{cov}(\mathbf{q_m},\mathbf{q_2}) & \dots & \mathbf{cov}(\mathbf{q_m},\mathbf{q_m}) \end{bmatrix} \end{aligned} \tag{A.9}$$

where

$$\mathbf{cov}(\mathbf{q_i}, \mathbf{q_j}) = \frac{\sum_{i=1}^{m}(\mathsf{q}_i - \mu_i)(\mathsf{q}_j - \mu_j)}{m-1}. \tag{A.10}$$

Calculate the eigenvectors and eigenvalues of the covariance matrix: An eigenvector of the covariance matrix $\mathbf{A}$ is a nontrivial vector $\mathbf{v}$ such that

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \tag{A.11}$$

for some scalar λ. And, λ is the corresponding eigenvalue. Let $\mathbf{I}$ be an identity matrix that has compatible size with $\mathbf{A}$ and $\mathbf{v}$. Equation (A.11) can be rearranged as follow:

$$\mathbf{A}\mathbf{v} - \lambda\mathbf{I}\mathbf{v} = 0, \tag{A.12}$$

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0. \tag{A.13}$$

For a non-zero eigenvector $\mathbf{v}$ to satisfy equation (A.13), the condition:

$$\mathbf{det}(\mathbf{A} - \lambda\mathbf{I}) = 0 \tag{A.14}$$

or

$$\begin{vmatrix} (a_{11}-\lambda) & a_{12} & \dots & a_{1m} \\ a_{21} & (a_{22}-\lambda) & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & (a_{mm}-\lambda) \end{vmatrix} = 0 \tag{A.15}$$

must be held. So, the eigenvalues $\lambda_i | i = 1 \dots m$ can be solved from this polynomial equation:

$$c_m\lambda^m + c_{m-1}\lambda^{m-1} + \dots + c_0 = 0. \tag{A.16}$$

A corresponding eigenvector $\mathbf{v}$ of each eigenvalue λ can be found by substitute a value of λ into equation (A.11).

Sort the feature vectors: For $i = 1 \dots m$ each of eigenvalue λ_i, there is a corresponding eigenvector (feature vector) $\mathbf{v}_{\lambda_i}$. To construct an eigenspace $\mathbf{V}$ for PCA mapping, the eigenvectors $\mathbf{v}_{\lambda_i}$ must be sorted such that $\lambda_1 > \lambda_2 > \dots \lambda_m$. Thus, the eigenspace can be expressed as:

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_{\lambda_1} & \mathbf{v}_{\lambda_2} & \dots & \mathbf{v}_{\lambda_m} \end{bmatrix}. \tag{A.17}$$

Transform the normalized data to the feature space: Let

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} \tag{A.18}$$

be the data in the feature space (eigenspace). Once the feature space is formed, the data can be transformed into the feature space by:

$$\mathbf{P} = \mathbf{V}^T\mathbf{Q} \tag{A.19}$$

when $\mathbf{V}^T$ is transpose matrix of $\mathbf{V}$. Each row of $\mathbf{P}$ contains significance of data in decreasing order. To transform the dimensions high-dimensional data $\mathbf{Q}$ to an dimensions low-dimensional data $\mathbf{X}$, let l be the number of dimension of $\mathbf{X}$ where $l < m$. The feature data $\mathbf{X}$ can be expressed in term of submatrix of $\mathbf{P}$ as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{l1} & p_{l2} & \cdots & p_{ln} \end{bmatrix}. \tag{A.20}$$

Suppose that the low-dimensional data of interest has three dimensions ($l = 3$). The feature data will be:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ x_{31} & x_{32} & \dots & x_{3n} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ p_{31} & p_{32} & \cdots & p_{3n} \end{bmatrix}. \tag{A.21}$$

Inverse mapping Let inverse mapping data $\widetilde{\mathbf{Q}}$ be an estimation of the normalized data $\mathbf{Q}$. Inverse PCA mapping from the low-dimensional space back to the original high-dimensional space can be done by:

$$\begin{aligned} \widetilde{\mathbf{Q}} &= (\mathbf{V}^T)^{-1}\widetilde{\mathbf{P}} \\ &= \begin{bmatrix} \tilde{q}_{11} & \tilde{q}_{12} & \cdots & \tilde{q}_{1n} \\ \tilde{q}_{21} & \tilde{q}_{22} & \cdots & \tilde{q}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{q}_{m1} & \tilde{q}_{m2} & \cdots & \tilde{q}_{mn} \end{bmatrix} \end{aligned} \tag{A.22}$$

where

$$\widetilde{\mathbf{P}} = \begin{bmatrix} \mathbf{X} \\ \hline \mathbf{O} \end{bmatrix} \tag{A.23}$$

and

$$\mathbf{O} = \begin{bmatrix} 0_{11} & 0_{12} & \dots & 0_{1n} \\ 0_{21} & 0_{22} & \dots & 0_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{(m-l)1} & 0_{(m-l)2} & \dots & 0_{(m-l)n} \end{bmatrix}. \tag{A.24}$$

The zero matrix $\mathbf{O}$ is combined to the low-dimensional data $\mathbf{X}$ to make the size of matrix $\widetilde{\mathbf{P}}$ to be compatible in equation (A.22). To converse $\widetilde{\mathbf{Q}}$ back to the original data space, a reverse normalization process also has to be performed by:

$$\widetilde{\mathbf{\Theta}} = \begin{bmatrix} \tilde{\theta}_{11} & \tilde{\theta}_{12} & \dots & \tilde{\theta}_{1n} \\ \tilde{\theta}_{21} & \tilde{\theta}_{22} & \dots & \tilde{\theta}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\theta}_{m1} & \tilde{\theta}_{m2} & \dots & \tilde{\theta}_{mn} \end{bmatrix} \tag{A.25}$$

where

$$\tilde{\theta}_{ij} = \tilde{q}_{ij}\sigma_i + \mu_i. \tag{A.26}$$

In the case of data in the feature space $\mathbf{X}$ has the same dimension with $\mathbf{Q}$ or in the case of $l = m$.

$$\widetilde{\mathbf{\Theta}} = \mathbf{\Theta} \tag{A.27}$$

will be the result. If $l < m$, some accuracy of the data will be lost.

Bibliography

[1] Rajesh P. N. Rao, Aaron P. Shon, and Andrew N. Meltzoff. *Imitation Learning inInfants and Robots: Towards probabilistic Computational Models.* Cambridge University Press, 2004.

[2] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[3] Miomir Vukobratović and Stepanenko Yu. On the stability of anthropomorphic systems. In *Mathematical Biosciences*, volume 15, pages 1–37, October 1972.

[4] Marion Sobotka, Dirk Wollherr, and Martin Buss. A jacobian method for online modification of precalculated gait trajectories. In *Proceedings of the 6th International Conference on Climbing and Walking Robots*, pages 435–442, Catania, Italy, 2003.

[5] D. Wollherr, M. Buss, M. Hardt, and O. von Stryk. Research and development towards an autonomous biped walking robot. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics AIM2003*, pages 968–973, Kobe, Japan, 2003.

[6] Shuuji Kajita, Osamu Matsumoto, and Muneharu Saigo. Real-time 3d walking pattern generation for a biped robot with telescopic legs. In *ICRA*, pages 2299–2306. IEEE, 2001. ISBN 0-7803-6578-X.

[7] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *ICRA* DBL [40], pages 1620–1626.

[8] Jun Morimoto, Sang-Ho Hyon, Christopher G. Atkeson, and Gordon Cheng. Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction. In *2008 IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, 2008, Pasadena, California, USA*, pages 2711–2716, 2008.

[9] Stuart Anderson, Martijn Wisse, Chris Atkeson, Jessica K Hodgins, Garth Zeglin, and B. Moyer. Powered bipeds based on passive dynamic principles. In *5th IEEE-RAS International Conference on Humanoid Robots, Tsukuba, Japan*, pages 110–116, December 2005.

[10] Jun Morimoto, Jun Nakanishi, Gen Endo, Gordon Cheng, Christopher G. Atkeson, and Garth Zeglin. Poincaré-map-based reinforcement learning for biped walking. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 18-22, 2005, Barcelona, Spain*, pages 2381–2386, 2005.

[11] Jun Morimoto, Gen Endo, Jun Nakanishi, Sang-Ho Hyon, Gordon Cheng, Darrin C. Bentivegna, and Christopher G. Atkeson. Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA*, pages 1579–1584, 2006.

[12] Koji Tatani and Yoshihiko Nakamura. Dimensionality reduction and reproduction with hierarchical nlpca neural networks-extracting common space of multiple humanoid motion patterns. In *ICRA* DBL [40], pages 1927–1932.

[13] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *Journal of the American Institute of Chemical Engineers*, 37(2): 233–243, 1991.

[14] Jack Wang, David Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1441–1448. MIT Press, Cambridge, MA, 2006.

[15] John Demiris and Gillian Hayes. A robot controller using learning by imitation. In *Proceedings of the 2nd International Symposium on Intelligent Robotic Systems*, Grenoble, France, 1994.

[16] Aude Billard. Imitation: a means to enhance learning of a synthetic protolanguage in autonomous robots. pages 281–310, 2002.

[17] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Trajectory formation for imitation with nonlinear dynamical systems. In *Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 752–757, 2001.

[18] Tetsunari Inamura, Iwaki Toshima, and Yoshihiko Nakamura. Acquisition and embodiment of motion elements in closed mimesis loop. In *ICRA* DBL [41], pages 1539–1544. ISBN 0-7803-7273-5.

[19] Michael J. Kirby and Rick Miranda. Circular nodes in neural networks. *Neural Comput.*, 8(2):390–402, 1996. ISSN 0899-7667.

[20] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Netw.*, 3(1):23–43, 1990. ISSN 0893-6080.

[21] Karl F. MacDorman. Feature learning, multiresolution analysis, and symbol grounding. In *Behavioral and Brain Sciences*, volume 21(1), pages 32–33, 1998.

[22] Y. Takahashi, M. Asada, and M. Asada. Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 1, pages 395–402 vol.1, 2000.

[23] William D. Smart and Leslie Pack Kaelbling. Practical reinforcement learning in continuous spaces. In Pat Langley, editor, *ICML*, pages 903–910. Morgan Kaufmann, 2000. ISBN 1-55860-707-2.

[24] Y. Takahashi, K. Hikita, and M. Asada. Incremental purposive behavior acquisition based on self-interpretation of instructions by coach. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 686–693 vol.1, 27-31 Oct. 2003.

[25] Vladimir M. Zatsiorsky. *Kinematics of Human Motion*. Human Kinetics Publishers; 1 edition, Champaign, IL, 1 edition, September 1997.

[26] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[27] Lawrence K. Saul and Sam T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifold. *Journal of Machine Learning Research*, 4: 119–155, 2003.

[28] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000. ISSN 0036-8075. doi: 10.1126/science.290.5500.2319. URL http://dx.doi.org/10.1126/science.290.5500.2319.

[29] E.C. Malthouse. Limitations of nonlinear pca as performed with generic neural networks. *Neural Networks, IEEE Transactions on*, 9(1):165–173, Jan 1998. ISSN 1045-9227. doi: 10.1109/72.655038.

[30] Richard Bellman. *Dynamic Programming*. Princeton University Press, June 1957.

[31] Rawichote Chalodhorn, Karl F. MacDorman, and Minoru Asada. An algorithm that recognizes and reproduces distinct types of humanoid motion based on periodically-constrained nonlinear pca. In Daniele Nardi, Martin Riedmiller, Claude Sammut, and José Santos-Victor, editors, *RobuCup*, volume 3276 of *Lecture Notes in Computer Science*, pages 370–380. Springer, 2004. ISBN 3-540-25046-8.

[32] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popovic. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

[33] Masaki Ogino, Yutaka Katoh, Masahiro Aono, Minoru Asada, and Koh Hosoda. Reinforcement learning of humanoid rhythmic walking parameters based on visual information. *Advanced Robotics*, 18(7):677–697, 2004.

[34] Olivier Michel. Webots: Symbiosis between virtual and real mobile robots. In *VW '98: Proceedings of the First International Conference on Virtual Worlds*, pages 254–263, London, UK, 1998. Springer-Verlag. ISBN 3-540-64780-5.

[35] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear systems part i: deterministic non-linear systems. *International Journal of Control*, 41(2):303–328, 1985.

[36] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear systems part ii: stochastic non-linear systems. *International Journal of Control*, 41 (2):329–344, 1985.

[37] David B. Grimes, Rawichote Chalodhorn, and Rajesh P. N. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In Gaurav S. Sukhatme, Stefan Schaal, Wolfram Burgard, and Dieter Fox, editors, *Robotics: Science and Systems*. The MIT Press, 2006. ISBN 0-262-69348-8.

[38] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989. ISSN 0005-1098.

[39] Shuuji Kajita, Takashi Nagasaki, Kenji Kaneko, Kazuhito Yokoi, and Kazuo Tanie. A running controller of humanoid biped hrp-2lr. In *ICRA*, pages 616–622, 2005.

[40] *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14-19, 2003, Taipei, Taiwan*, 2003. IEEE.

[41] *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, 2002. IEEE. ISBN 0-7803-7273-5.