# Automatic extraction of abstract actions from humanoid motion data

Rawichote Chalodhorn[1], Karl MacDorman[1, 2], and Minoru Asada[1, 2]
[1]Department of Adaptive Machine Systems
[2]Handai Frontier Research Center
Graduate School of Engineering, Osaka University
Yamadaoka 2-1, Suita, Osaka 565-0871 Japan
chote@er.ams.eng.osaka-u.ac.jp

***Abstract*—Developing a humanoid robot that can learn to perform complex tasks by itself has become a major goal of robotics research. This paper proposes a new algorithm for the automatic segmentation of humanoid motion data. We use a simplified soccer game, RoboCup, as a prototype task. Motion data from a 20 degree-of-freedom humanoid soccer playing robot are reduced to their intrinsic dimensionality by nonlinear principal component analysis. The proposed algorithm operates in of two phases. The first phase automatically segments the motion data in the reduced sensorimotor space by incrementally generating nonlinear principal component analysis with a circular constraint networks and assigning data points based on their temporal order to these networks in a conquer-and-divide fashion. Then, the second phase of the algorithm removes repeated patterns based on the distance between redundant motion patterns in the reduced sensorimotor space. The networks abstracted five motion patterns without any prior information about the number or type of motion patterns. The learned networks can be used to recognize and generate humanoid actions.**

***Keywords: Humanoid robot; pattern recognition; automatic segmentation; nonlinear principal component analysis***

## I. INTRODUCTION

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 some simple tasks. However, such demonstrations are usually performed 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 these conditions [1]. One solution is to develop a robot that can learn to perform in a human environment by itself.

Reinforcement learning [2] provides a useful method of adapting to environmental change based on experience. The self-organized, modular, and hierarchical structure of multi-layered reinforcement learning [3] extends reinforcement learning to more complicated learning problems. 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 [4]. A heuristic algorithm is applied to a sample set to generate a state-action space and learning modules automatically [5]. The learning modules [5] are also reusable for learning new complex behavior. The reinforcement learning method works well with a simple robot, such as a wheeled robot. 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 [5], instead of using a raw state-action space.

We propose a new algorithm that segments humanoid motion data automatically. The segmentation results can be used as abstract states and abstract actions to facilitate the learning of complex tasks by the hierarchical multi-module reinforcement learning method [5]. We used nonlinear principal component analysis (NLPCA) [6] to reduce the highly dimensional space of humanoid motion data to a tractable three-dimensional feature space. Our algorithm then incrementally employs nonlinear principal component analysis with a circular constraint (CNLPCA) networks [7] to learn and divide data into segments. A CNLPCA 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 network is employed. The algorithm keeps exploiting CNLPCA networks in the temporal ordering of the data 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 exist in the result. Our 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. In the final results, all the periodic motion patterns are characterized by automatically segmented trajectories.

## II. RELATED WORKS

Linear PCA is a common method of dimensionality reduction. However, linear PCA has a problem representing nonlinear humanoid motion data. Tatani and

Nakamura [8] were first to apply NLPCA to human and humanoid motion data, though for dimensionality reduction only. A number of imitation frameworks have been proposed. A nonlinear dynamical system [9] was crafted to produce primitive behaviors. A framework that is based on human designed behaviors may lack of the essence of developing behaviors through embodiment [10]. The mimesis theory [11] proposed action acquisition and action symbol generation while considering the embodiment concept. However, action symbols in the mimesis theory are not automatically extracted from the sequence of motion data. A very similar framework to the work here, which uses dimensionality reduction and segmentation of motion data in the reduced sensorimotor data space [12], also does not segment the motion data automatically.

## III. NONLINEAR PRINCIPAL COMPONENT ANALYSIS WITH A CIRCULAR CONSTRAINT

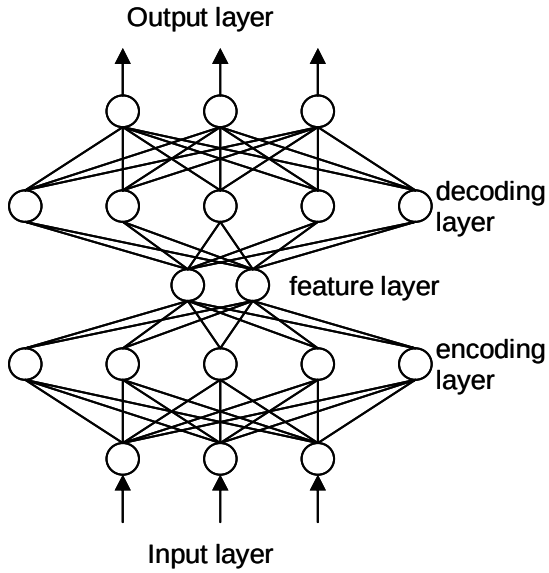


Figure 1. Target values presented at the output layer of a nonlinear principal component neural network are identical to input values. Nonlinear units comprise the encoding and decoding layers, while either linear or nonlinear units comprise the feature and output layers.

The human body has 244 degrees of freedom [13] and a vast array of proprioceptors. Excluding the hands, a humanoid robot generally has at least 20 degrees of freedom and far more dimensions are required to describe its dynamics precisely. Fortunately, from the standpoint of a particular activity, the effective dimensionality may be much lower.

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

$$\left\| x - g\left(f(x)\right)\right\|^2, \quad x \in \mathbf{R}^N,$$

resulting in $P$ principal components $\lfloor y_1 \ldots \quad y_p \rfloor = f(x)$ in the feature layer. Kramer [6] first solved this problem by training a multilayer perceptron as shown in Fig. 1 using backpropagation of error, although a second order method such as conjugate gradient analysis converges to a solution faster for many large data sets.

NLPCA unlike PCA, which is a special case where $C$ and $D$ are linear, does not have a unique solution and no known computational method is guaranteed to find any globally optimal solution. However, NLPCA can address the nonlinear nature of humanoid motion data better than PCA. Moreover, NLPCA also provides a reverse mapping and interpolation from the feature space back to the original data space of high dimensionality.

Here, we use NLPCA for dimensionality reduction. The performance of NLPCA for dimensionality reduction is acceptable. From our preliminary investigation of the humanoid motion patterns in the feature space of NLPCA, most of the patterns are closed curves because of their periodic nature. Conventional NLPCA has a problem with learning a closed or self-intersecting curve [14], while nonlinear principle component analysis with a circular constraint at the feature layer (CNLPCA) can overcome this difficulty [7]. To represent these closed curves, we instead use CNLPCA to learn the periodic motion patterns in the feature space.

Kirby and Miranda [7] constrained the activation values of a pair of nodes $\mathbf{p}$ and $\mathbf{q}$ in the feature layer of an NLPCA network to fall on the unit circle:

$$r = \sqrt{\left(p_0^2 + q_0^2\right)},$$

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

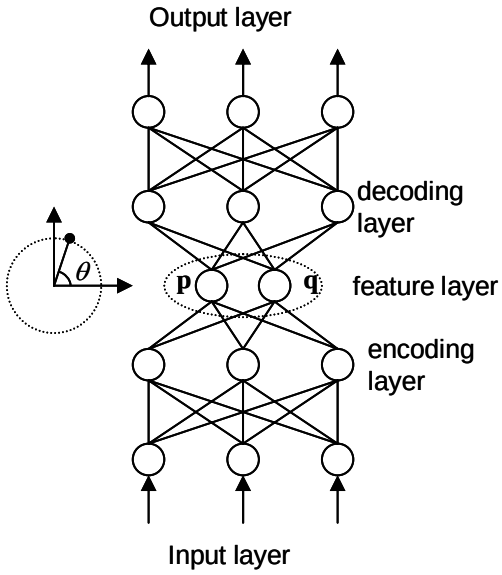


Figure 2. The NLPCA with a circular constrain at the bottleneck.

While $p_0$ and $q_0$ are the input activation, $p$ and $q$ are the output of nodes **p** and **q**, respectively. Thus, the pair of nodes **p** and **q** acting as a single angular variable $\theta$.

## IV. Automatic Segmentation Algorithm

We conceived of the automatic segmentation problem as the problem of uniquely assigning a temporal sequence of data points in the feature space to CNLPCA 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 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. The automatic segmentation algorithm works as follows:

LIST I. Pseudocode 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 $MSE_{new} < (1+\alpha) \times MSE_{old}$ goto step 2.
5. End learning of this segment.
6. Goto step 1, if this is not the end of the data set.

As stated in List 1, the automatic segmentation begins to work by creating a new CNLPCA network. Then, a number of data points *n* in the data sequence are selected and assigned to the CNLPCA network that was created in the previous step. The number of data points *n* is not a critical free-parameter of our algorithm, *n* could be any positive integer number starting from 1. In the other word, the parameter *n* is the size of the new data set that is added to the network to learn a pattern at each iteration of the algorithm. Thus, if we increase *n*, the iteration step in List 1 will be decreased. However, we should not use a large value for *n*. Because, if the value of *n* is close to the total number of data points in a segment, that means the segmentation is biased by the parameter *n*. After *n* data points have been assigned to the network, the network training begins. In this work, the terminal criterion of network learning is 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 the most important step of the automatic segmentation algorithm, because the decision to continue learning on the same segment or to begin a new segment is made at this step. The decision to continue learning by using the same network is made by comparing the value of mean square error of the network before and after the network is assigned to learn the *n* data points. A free parameter $\alpha$ is used in the comparison. The parameter $\alpha$ is a small positive real number that is less than 1. We can interpret the meaning of the parameter $\alpha$ as the factor that indicates the allowance of increasing of the mean square error value of the learning network when a new set of *n* data points are assigned to it. The comparison condition of the if-statement in step 4 of List 1 that allows increasing of the mean square error value, does not lead to a large value of the mean square error at the end of segment learning, because the mean square error value will be decreased again on the next iteration of the learning of the network. If the mean square error value can not be decreased and its value exceeds the allowance 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 networks and assigning *n* data points to them until the end of the data is reach.

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. We want to obtain only one network for each abstract motion pattern. Thus, the redundant networks should be removed or at least minimized. The minimization of networks redundancy is performed by the following steps:

LIST II. Pseudocode for Minimization of Networks Redundancy

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

To obtain a value of the average distance $d_{avg}$ between segment *a* and segment *b*, one may calculate output of network *a* and network *b* by running the angular parameter at the bottleneck layer of network from 0 to 2π, then uses the output of network *a* as the input data to network *b*, the average value between input-output pairs of the network *b* is $d_{avg}$. The inverse of the square of average distance $D_j$ is used for a clear discrimination between segments that are close or far from the reference segment.

## V. Experimental Results and Discussion

This section shows the result of automatic segmentation. We assess the accuracy of the result based on a manual segmentation of the data and an analysis of how data points are allocated among the networks. The segmentation results before and after applying minimization of networks redundancy are also shown here.

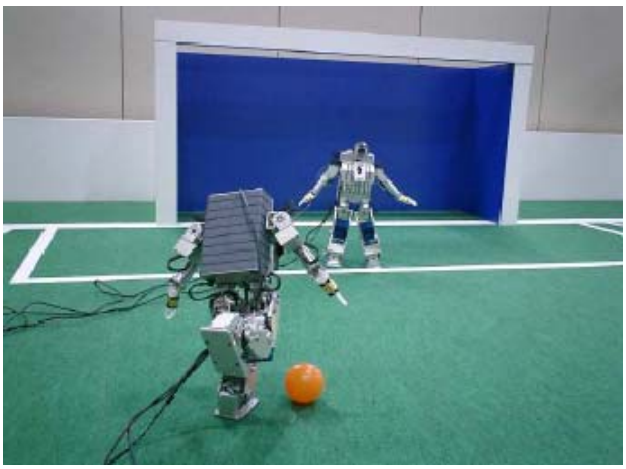

Figure 3. The Fujitsu HOAP-1 robots are playing a simplified soccer game: RoboCup.

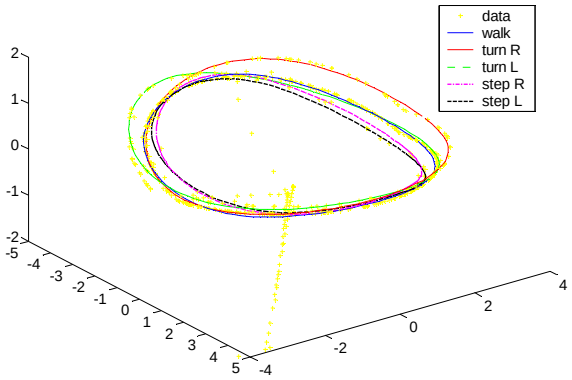


Figure 4. Recognized motion patterns embedded in the dimension of the first three nonlinear principal components of the raw proprioceptive data.

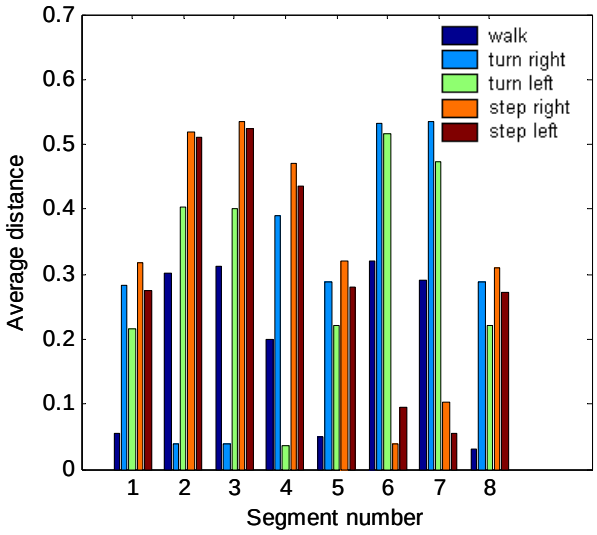


Figure 5. The average distance between manually segmented networks and automatically segmented networks before eliminating redundant networks.

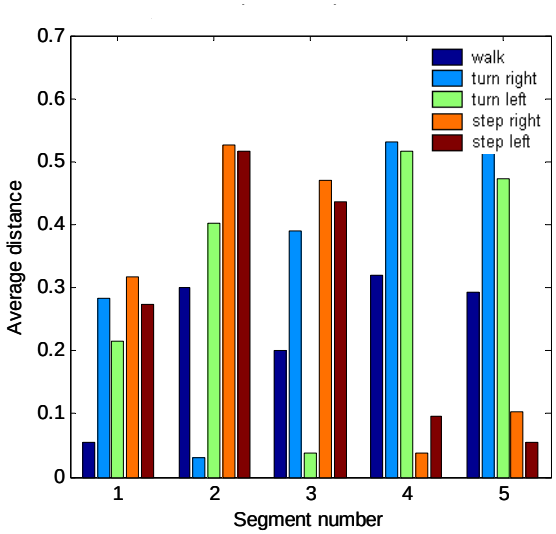


Figure 6. The average distance between manually segmented networks and automatically segmented networks after eliminating redundant networks.

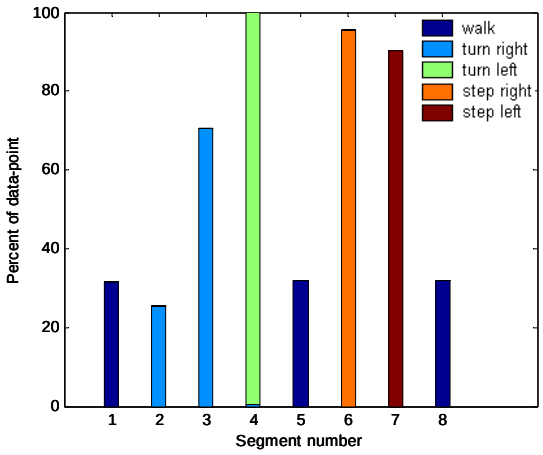


Figure 7. The allocation of data points to each network before applying network redundancy minimization.

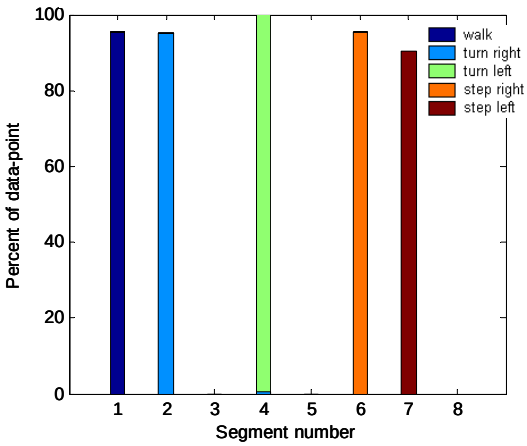


Figure 8. The allocation of data points to each network after applying network redundancy minimization.

We recorded motion data while a Fujitsu HOAP-1 humanoid robot was manually controlled by a human operator to play soccer as shown in Fig. 3. The motion sequences are walking forward, turning right, turning left, walking forward, sidestepping right, sidestepping left, and kicking. Each data point is constituted by a 20-dimension vector of joint angles. A standard (noncircular constraint) NLPCA network reduced the dimensionality of the data from 20 to 3.

As explained in the previous section, our algorithm consists of two phases: automatic segmentation of motion data in the data sequence and minimization of networks redundancy based on average spatial distance between segments in the reduced sensorimotor space. We have obtained eight segments of motion data patterns after we performed the automatic segmentation based on the temporal order of the data. An accuracy analysis of the segmentation results based on a manual segmentation which is the average distances between manually segmented trajectories and automatically segmented trajectories is shown in Fig. 5. 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 Fig. 7. After the automatic segmentation routine has been performed, the routine for minimizing redundant networks searches for segments those positions are very close to each other and merges them. Fig. 4 shows that the complete automatic segmentation routine successfully employed CNLPCA networks to separate and generalize five of the periodic motions without any prior information about the number or type of motion patterns.

Fig. 5 and 6 are analyses of average distances from each automatically segmented pattern to every 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 minimization of networks redundancy algorithm as shown in Fig. 5. The lowest bar indicates which known pattern matches the automatically segmented pattern. We notice from Fig. 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 Fig. 5, are also redundant. Both represent the turning right action. This is an inaccurate result, 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. The allowance factor of increasing of the mean square error value of the learning network parameter $\alpha$ , influences these results. The smaller value of $\alpha$ we use, the more segments we get. Even a motion pattern could be broken into several segments at this state, but segments that represent the same motion pattern will be merged later by the minimization of networks redundancy routine.

An underlying fact that allows the minimization of networks redundancy routine to search and merge redundant and broken segments that represent the same motion pattern is the segments that represent the same motion pattern have the same shape of curve and they lie near each other in the reduced sensorimotor space. All of the redundant networks were removed and their data points were reallocated. Fig. 7 and 8 are an analysis of the allocation of data points before and after applying the minimization of networks redundancy algorithm, respectively. 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 point of each of pattern in a segment to the total number of data points of each pattern in the entire data set. We observe a very low rate of data point misallocation in Fig. 7. The allocation of data points after the removal of the redundant networks is also accurate. We can observe from Fig. 7 that segment No. 5 and 8, which are redundant with respect to segment No. 1, were merged into segment No. 1 in Fig. 8. Segment No. 3 which is redundant with segment No. 2, was also merged into segment No. 2 in Fig 8.

However, our algorithm could not capture the kicking pattern. This is because it appears to be a very irregular discontinuous curve in the feature space. We plan to fix this problem by using a more powerful functional approximation algorithm.

## VI. Conclusion

We proposed an automatic segmentation algorithm for humanoid motion data. The algorithm is designed for working with motion data that was recorded from a real humanoid task. The algorithm abstracted five out of six types of humanoid motion without any prior information about the number or type of motion patterns.

Our 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 sensorimotor trajectories they generate. Our algorithm can perform well for periodic humanoid motion patterns. The learned networks can be used for recognition and generation of humanoid actions in order to learn and perform a complex task as in [5].

These results can facilitate the learning of human's complex tasks by deriving an abstract state-action space for reinforcement learning. However, we have not included a robot's dynamic information in this work. Thus, instability dynamics could result, if the demonstrator and the learner have significant differences in their dynamics. We plan to include the robot's dynamic information in our future work.

## References


[1] K. F. MacDorman, "Feature learning, multiresolution analysis, and symbol grounding," Behavioral and Brain Sciences, vol. 21, pp. 32-33, 1998.

[2] R. Sutton and A. Barto, An Introduction to Reinforcement Learning, MIT Press, 1998.

[3] Y. Takahashi and M. Asada, “Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning,” IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000, pp. 395-402.

[4] W. D. Smart and L. P. Kaelbling, “Practical reinforcement learning in continuous spaces,” Proceedings of the Seventeenth International Conference on Machine Learning, 2000, pp. 903-910.

[5] Y. Takahashi, K. Hikita, and M. Asada, “Incremental purposive behavior acquisition based on self-interpretation of instructions by coach,” IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003, pp. 686-693.

[6] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” Neural Computation, vol. 9, pp. 1493-1516, 1991.

[7] M. J. Kirby and R. Miranda, “Circular nodes in neural networks,” Neural Computation, vol. 8 , pp. 390-402, 1996.

[8] M. Okada, K. Tatani, Y. Nakamura, “Polynomial design of the nonlinear dynamics for the brain-like information processing of the whole body motion,” IEEE International Conference on Robotics and Automation, 2002, pp.1410-1415.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Trajectory formation for imitation with nonlinear dynamical systems,” IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001, pp.752-757.

[10] K. F. MacDorman, “Grounding symbols through sensorimotor integration,” Journal of the Robotics Society of Japan, vol. 17, pp.20-24, 1999

[11] T. Inamura, I. Toshima, and Y. Nakamura, “Acquiring motion elements for bi-directional computation of motion recognition and generation,” In Siciliano, B., Dario, P., Eds., Experimental Robotics VIII, Springer, pp. 372-381, 2003.

[12] O. C. Jenkins and M. J. Mataric, “Automated derivation of behavior vocabularies for autonomous humanoid motion,” Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, 2003, pp. 225-232.

[13] V. M. Zatsiorsky, “Kinematics of human motion,” Human Kinetics, Urbana Champaign, 2002.

[14] E. C. Malthouse, “Limitations of nonlinear PCA as performed with generic neural neworks,” IEEE Transactions on Neural Networks, vol. 9, pp. 165-73, 1998.