Humanoid Robot Motion Recognition and Reproduction

Rawichote Chalodhorn[1] Karl F. MacDorman[2] Minoru Asada[3]

[1] *Neural Systems Laboratory, Department of Computer Science and Engineering University of Washington Seattle, WA 98195-2350 USA Tel.: +1-206-685-3806 Fax: +1-206-543-2969 E-mail: choppy@cs.washington.edu*

[2] *School of Informatics, Indiana University, Indianapolis, IN 46202 USA Tel.: +1 317 215-7040 Fax.: +1 206 350-6089 URL: macdorman.com*

[3] *Department of Adaptive Machine Systems, Graduate School of Engineering, Osaka University Suita Osaka 565-0871 Japan Tel.: +81-6-6879-7347 Fax: +81-6-6879-7348 E-mail: asada@ams.eng.osaka-u.ac.jp*

Abstract

Humanoid robots have become appealing to the research community because of their potential versatility. However, traditional programming approaches may not reveal their full capabilities. Thus, an important goal is to develop a humanoid robot that can learn to perform complex tasks by itself. This paper proposes a method to recognize and regenerate motion in a humanoid robot. We demonstrate how a sequence of high-dimensional motion data can be automatically segmented into abstract action classes. 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 (NLPCA). The motion data is then segmented automatically by incrementally generating NLPCA networks with a circular constraint and assigning to these networks data points according to their temporal order in a conquer-and-divide fashion. Repeated motion patterns are removed based on their proximity to similar motion patterns in the reduced sensorimotor space to derive a nonredundant set of abstract actions. The networks abstracted five motion patterns without any prior information about the number or type of motion patterns. A motion optimization algorithm ensured motion reproduction by employing the sensorimotor mapping in the low-dimensional space.

Keywords: Automatic segmentation; humanoid robot; motion learning; nonlinear principal component analysis; pattern recognition

1 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 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 [10]. One solution is to develop a robot that can learn to perform in a human environment.

Reinforcement learning [17] 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 [18]. 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 [16]. 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 [19]. 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 [19] instead of using the raw state-action space determined by the sensors and effectors.

Principal components analysis (PCA) is a common *linear* method of dimensionality reduction. However, PCA has a problem representing nonlinear humanoid motion data. Tatani et al. [14] were first to apply nonlinear principal components analysis to human and humanoid motion data, though for dimensionality reduction only. A number of imitation frameworks have been proposed. A nonlinear dynamical system [4] was crafted to produce primitive behaviors. A framework that is based on human-designed behaviors may lack the flexibility of behaviors developed through embodiment [11]. The mimesis theory proposed action acquisition and action symbol generation while accounting for embodiment [5]. 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 a sensorimotor space of reduced dimensionality [6], also does not segment the motion data automatically.

We propose an approach 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 hierarchical multi-module reinforcement learning method [19]. We used nonlinear principal component analysis [8] to reduce the high-dimensional space of humanoid motion data to a tractable three-dimensional feature space. Our algorithm then incrementally employs nonlinear principal component neural networks with a circular constraint (CNLPCA) [7] 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 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. As a result, automatically segmented trajectories characterize all the periodic motion patterns. We also provide a method to regenerate dynamically stable motions by using the compact representation of motion data in the low-dimensional space to create a model of the casual relation between the motion commands and their subsequent sensor feedback. From this model, we are able to derive an optimal action for a dynamically balanced condition at each time step.

2 NONLINEAR PRINCIPAL COMPONENT ANALYSIS WITH A CIRCULAR CONSTRAINT

The human body has 244 degrees of freedom [21] 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. Fig. 1 illustrates a data pattern of a walking motion in a three-dimensional space after applying a dimension reduction algorithm to the high-dimensional joint angle data. The first two labeled postures in Fig. 1 are intermediate postures between an initial stable standing pose and points along the periodic gait loop represented by postures three through eight. Note the low-dimensional space preserves the temporal-order of the data set. This property increases the likelihood of regenerating the motion from the low-dimensional space.

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 :

$$\|\vec{x} - g(f(\vec{x}))\|^2, \quad \vec{x} \in \mathbb{R}^N \tag{1}$$

resulting in P principal components $[y_1 \cdots y_p] = f(\vec{x})$ in the feature layer. Kramer [8] first solved this problem by training a multilayer perceptron as shown in Fig. 2 using backpropagation of error, although a second order method such as conjugate gradient analysis converges to a solution faster for many large data sets. We used three-dimensional data at the feature layer, which were produced by an NLPCA neural network that has three nodes at the feature layer for all of the low-dimensional space representation throughout this paper.

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

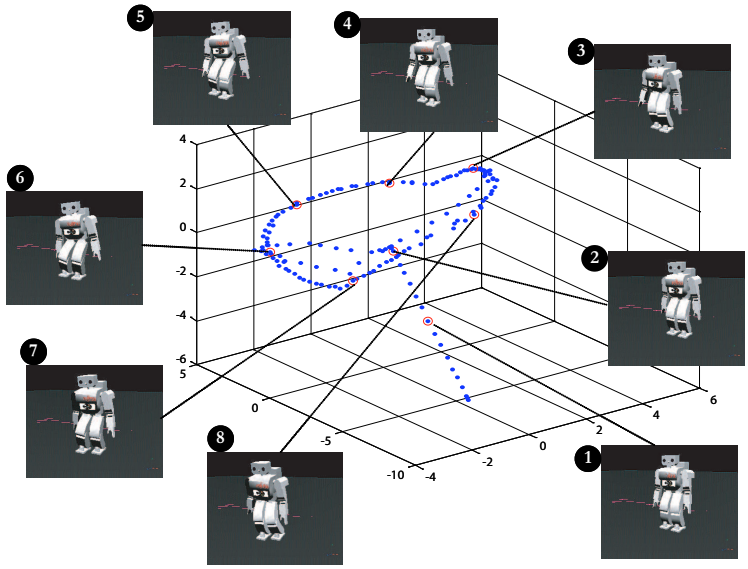

Figure 1: A three-dimensional space representation of the postures of the Fujitsu HOAP-2 robot. Blue points along a loop represent different robot postures during a single walk cycle.

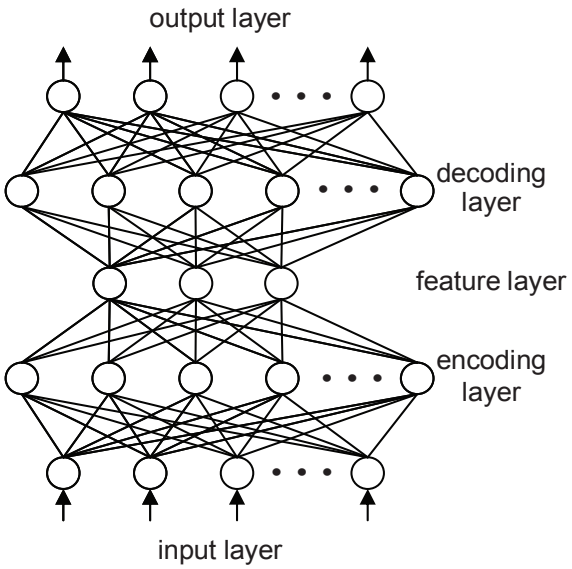

Figure 2: The structure of the nonlinear principal components network is symmetrical with respect to its feature layer. The number of input nodes and output nodes are set to the number of dimensions of the input data. Target values presented at the output layer are set to be identical to input values. 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.

for convergence. In 2008 Hinton and Salakhutdinov [3] found an effective way to initialize the weights of an NLPCA neural network. Unlike PCA and nonparametric methods such as [15], [20], NLPCA autoencoders give mappings in both directions between high-dimensional space and low-dimensional space.

This research uses NLPCA for dimensionality reduction, and its performance for that purpose 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 is unsuitable for learning a closed or self-intersecting curve [12]. However, nonlinear principle component neural networks with a circular constraint at the feature layer (CNLPCA) can overcome this difficulty [7]. To represent these closed curves, we 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 p and q in the feature layer of an NLPCA neural network to fall on the unit circle:

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

$$p = \frac{p_0}{r} \text{ and } q = \frac{q_0}{r}. \tag{3}$$

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

$$\theta = \arctan \frac{p}{q}. \tag{4}$$

3 AUTOMATIC SEGMENTATION ALGORITHM

We conceived of automatic segmentation 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:

From Table 1, the automatic segmentation begins to work by deploying a CNLPCA neural network. Then, a number of data points n in the data sequence in low-dimensional space are assigned to the network that was created in the previous step. The size of n is not a critical free-parameter of our

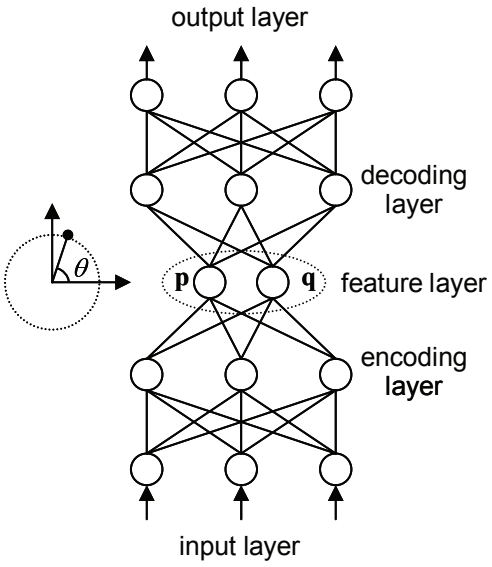

Figure 3: The NLPCA network with a circular constrain at the bottleneck layer. In this work, the number of nodes in the input layer, encoding layer, feature layer, decoding layer, and output layer of CNLPCA are 3, 3, 2, 3 and 3 respectively as depicted in this figure.

Table 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.

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 1. However, we should use a value for n that is 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 free parameter α is used in the

comparison. 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 have decreased 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 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. We would like for only one network to represent one abstract motion pattern. Thus, the redundant networks should be removed or at least reduced in number. The following steps minimize network redundancy:

Table 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.

4 MOTION RECOGNITION: AUTOMATIC SEGMENTATION

This section reports the results of automatic segmentation. We assess the accuracy of the results 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 reported.

We recorded motion data while a human operator manually controlled a Fujitsu HOAP-2 humanoid robot to play soccer, as shown in Fig. 4. 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 Fig. 5. These steps are data preprocessing[3] for more efficient automatic segmentation by the CNLPCA algorithm.

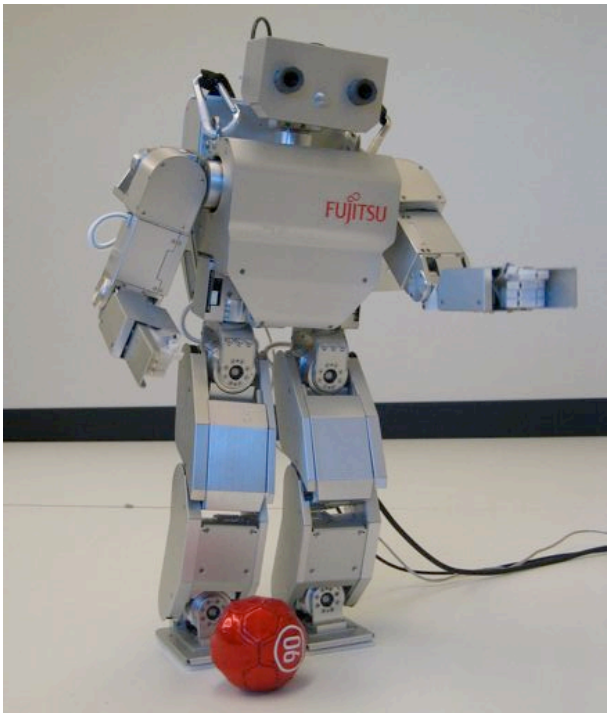

Figure 4: Fujitsu HOAP-2 robot's ball following behavior.

As explained in the previous section, our algorithm for motion recognition consists of two phases: automatic segmentation of motion data in the data sequence and network redundancy minimization based on average spatial distance between segments in the reduced sensorimotor space. We have obtained

[1]To demonstrate that our algorithm is able to handle redundant motion patterns, the walking forward motion intentionally appears twice 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 our motion patterns.

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

eight segments of motion data patterns after we performed the automatic segmentation according to the temporal order of the data. An accuracy analysis of the segmentation results is shown in Fig. 6. The figure compares the average distances 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 Fig. 8. After automatic segmentation has completed, the routine for redundant network minimization searches for segments whose positions are very close to each other and merges them. Fig. 5 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.

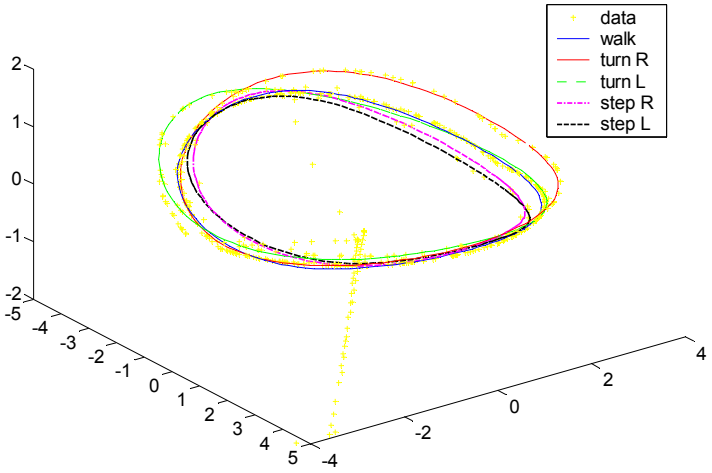

Figure 5: Recognized motion patterns embedded in the dimensions of the first three nonlinear principal components of the raw proprioceptive data.

One may calculate the average distance from a manually segmented data point to an automatically segmented data point by providing the manually segmented data point as input to the CNLPCA network and calculate the average value. Fig. 6 and 7 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 Fig. 6. The lowest bar indicates which known pattern matches the automatically segmented pattern. We notice from Fig. 6 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. 6 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.

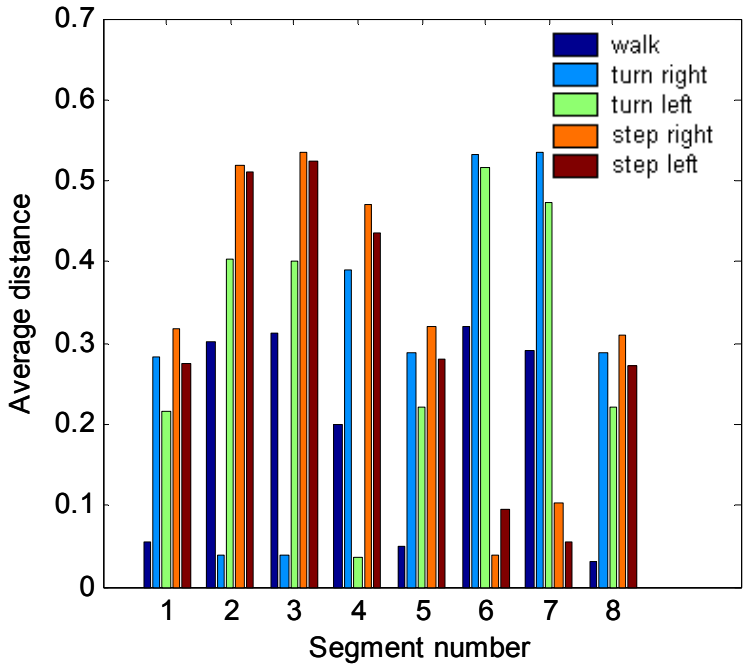

Figure 6: The average distance between manually and automatically segmented neural networks before eliminating redundant networks. (A shorter bar indicates greater similarity with respect to the reference pattern.)

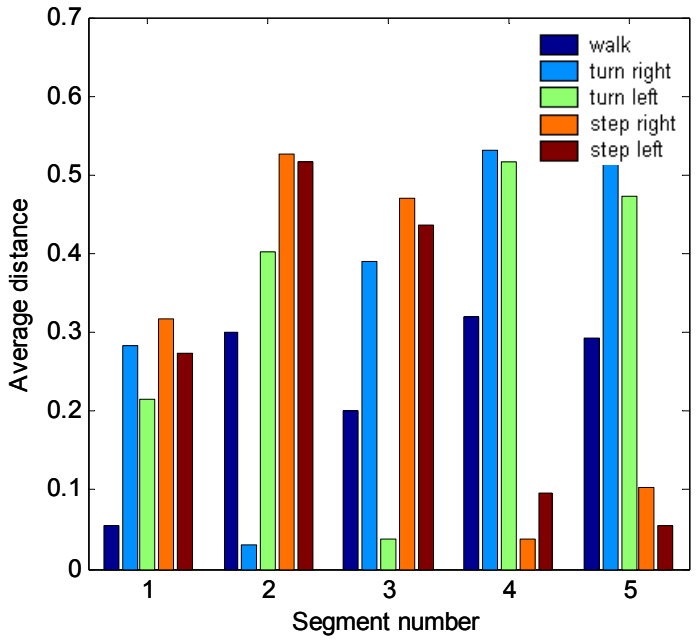

Figure 7: The average distance between manually and automatically segmented neural networks after eliminating redundant networks. (A shorter bar indicates higher similarity with respect to the reference pattern.)

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.

The routine that searches for redundant and partial segments that represent the same abstract action and merges them is able to achieve this because the segments have a similar curvature and lie near each other in the reduced sensorimotor space. All of the redundant networks were removed and their data points were reallocated. Fig. 8 and 9 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 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. 8. The allocation of data points after the removal of the redundant networks is also accurate. We can observe from Fig. 8 that segment No. 5 and 8, which are redundant with respect to segment No. 1, were merged into segment No. 1 in Fig. 9. Segment No. 3, which is redundant with respect to segment No. 2, was also merged into segment No. 2 in Fig. 9.

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.

Barbic et al. [1] have studied the automatic segmentation algorithm for motion capture data in parallel with our work. They defined the automatic segmentation precision of each algorithm as the ratio of reported correct cuts versus the total number of reported cuts.[4] They found that the precision of using principal components analysis, probabilistic principal components analysis (PPCA), and a Gaussian mixture model (GMM) for automatic segmentation are 0.79, 0.92 and 0.77, respectively. While our results cannot be compared directly with their results, because the data sets are different, an estimation of precision of CNLPCA by their method of using cuts is 0.88 before and 1.00 after network redundancy minimization.

5 MOTION REPRODUCTION BY SENSORIMOTOR MAPPING

In section 2 and Fig. 3, one may reproduce a motion pattern for a robot from a segmented motion pattern by giving input ranging from 0 to 2π for θ at the circularly constrained nodes of the CNLPCA

[4] A higher number indicates greater precision.

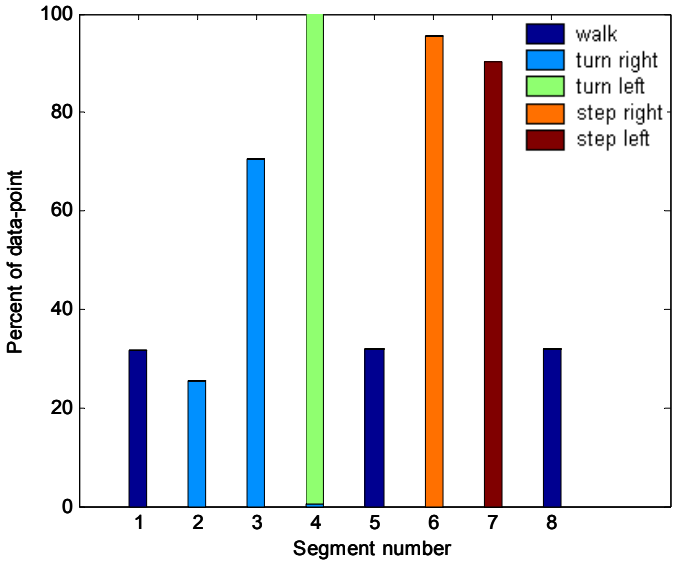

Figure 8: The allocation of data points to each network before applying network redundancy minimization.

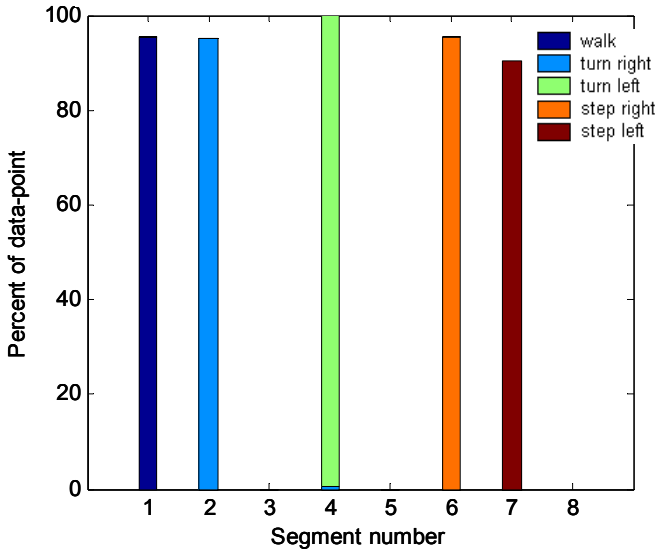

Figure 9: The allocation of data points to each network after applying network redundancy minimization.

feature layer. However, the resulting motion may be undesirable because of a loss in accuracy resulting from the decoding layers of the CNLPCA and NLPCA networks. In attempting to reproduce the motion on a different robot that has a kinematic structure similar to the one that produced the motion, slightly different dynamical properties of the robots may result in an even more undesirable motion. The desired motion can be obtained, if the reproduced motion is adjusted using embodied sensory feedback from a learning process. Thus, in this section we introduce a technique to optimize the sensorimotor mapping through the low-dimensional space.

For the walking forward motion, the desirable motion is a stable walking gait that makes the robot walk straight without any unexpected turns.[5] Thus, our objective is to search for a set of actions a_t^* that minimizes the expected states of *gyroscopic feedback*[6] during the motion:

$$a_t^* = \arg\min_{a_t} \Gamma(\tilde{\boldsymbol{\omega}}_{t+1}), \tag{5}$$

$$\Gamma(\tilde{\boldsymbol{\omega}}_{t+1}) = \lambda_x \tilde{\omega}_x^2 + \lambda_y \tilde{\omega}_y^2 + \lambda_z \tilde{\omega}_z^2. \tag{6}$$

In other words, at time t, we search for an optimal action a_t^* from the set of all possible actions a_t such that the objective function Γ in Eq. (6) is minimized. The objective function Γ represents the sum of squares of the predicted three-axes gyroscope signal of $\tilde{\omega}_x$, $\tilde{\omega}_y$, and $\tilde{\omega}_z$ of the next time step $t+1$. The expected gyroscope signals $\tilde{\boldsymbol{\omega}}_{t+1}$ is a predicted sensory state s_{t+1} where

$$s_{t+1} = F(s_t, s_{t-1}, ..., s_{t-n-1}, a_t, a_{t-1}, ..., a_{t-n-1}) \tag{7}$$

We assume that the casual relation between sensory feedback s_t and the motor command a_t is a stationary process. The model state predictor Eq. (7) is a Markovian function that estimates state s_{t+1} based on the current causal state-action pair and its historical information. In this paper we use a time-delay radial basis function (RBF) network [9] to learn the model state predictor. The optimization process in Eq. (5) can be rewritten in term of state and action:

$$a_t^* = \arg\min_{a_t} \Gamma(F(s_t, \ldots, s_{t-n}, a_t, \ldots, a_{t-n})). \tag{8}$$

The singular variable in Eq. (4) can be used as a constrained motor command through the decoder layers of CNLPCA and NLPCA, respectively. In this paper, we use the second-order state estimator. Eq. (8) becomes:

$$\theta_t^* = \arg\min_{\theta_t \in \Theta_s} \Gamma(F(\tilde{\boldsymbol{\omega}}_t, \tilde{\boldsymbol{\omega}}_{t-1}, \theta_t, \theta_{t-1})). \tag{9}$$

[5] An unexpected turn can occur in a walking gait from factors such as improper step timing or nonuniform friction with the ground.

[6] A three-channel gyroscope sensor was placed near the stationary standing center-of-mass of the robot.

$$\theta_{t-1} < \Theta_s \leq \theta_{t-1} + \epsilon_\theta \tag{10}$$

$$0 < \epsilon_\theta < 2\pi \tag{11}$$

At each sampling period, the search space Θ_s is a region of the estimated circular curve in the feature space defined by Eq. (10) and Eq. (11).

Selected actions will only be optimal if the sensorimotor predictor is accurate. We, therefore, periodically retrain the prediction model based on the new motor commands a_t^* generated by the optimization algorithm and the predicted sensory feedback obtained from executing these commands. After three iterations of sensorimotor prediction learning, a desirable walking gait is obtained. Fig. 10 shows a trajectory of the optimized walking gait in the feature space. We can see from Fig. 10 that the optimized moves skipped some regions and concentrated on other regions of the constraint pattern. The regions of the motion pattern that were skipped are those causing body oscillation. The algorithm reassigned walking postures to the parts of the motion pattern that produces lower oscillation.

We tested the optimization result using commercial dynamics simulation software [13]. The simulation results are shown in Fig. 11. For the optimized motion, the robot walks faster, more erect, and with fewer unexpected turns than the walking gait that is reproduced without optimization. The robot appears to swing its legs faster and stay longer in the double-support phase of the walking gait. This is consistent with the plot in Fig. 10 and consistent with our objective function in Eq. (6). The motion is not quasi-static but dynamically stable: The robot will fall, if we stop sending motion commands while it is walking. Although our method does not guarantee dynamic stability, it is minimizing the chance of the robot falling as learning progresses and enabling the robot to compensate for minor unexpected perturbations.

6 Conclusion

We have proposed a framework for motion recognition and reproduction in 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. Our 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 sensorimotor trajectories they generate.

Our algorithm performs well for periodic humanoid motion patterns. Note that, although the joint angle space used in this research is 20 dimensional, the proposed algorithm is not limited to working

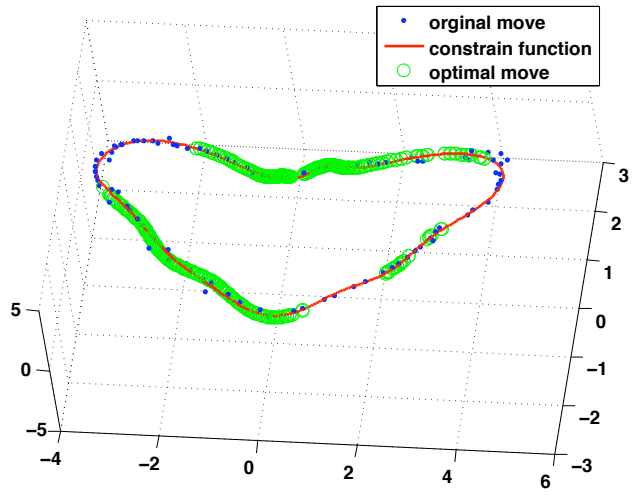

Figure 10: Motion-phase optimization. After three learning iterations of the sensorimotor causal relation, the optimization algorithm planned a sequence of actions in low-dimensional space based on its objective function in Eq. (9).

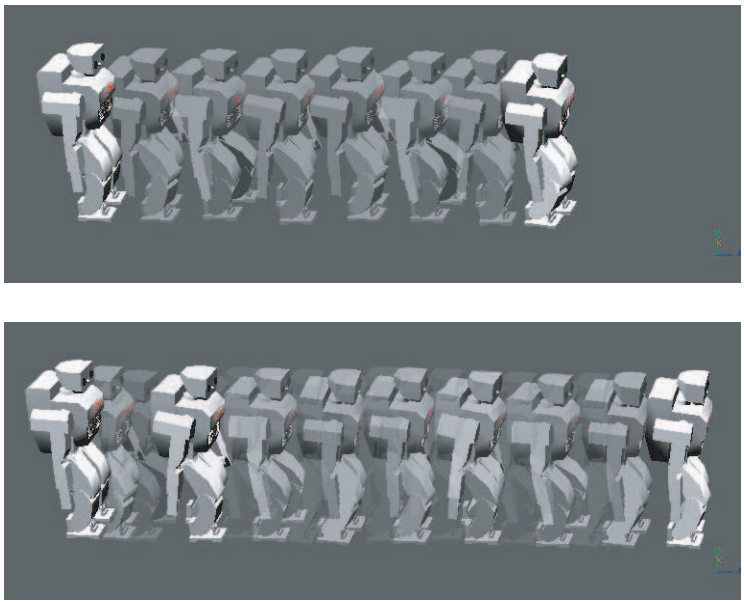

Figure 11: A comparison of the initial and optimized walking gait. The top figure depicts the robot performing the initial walking gait for two seconds. Motion-phase optimization results in a faster walking gait, as shown in the bottom figure.

with no more than 20 dimensions. However, the NLPCA network training time will increase as more dimensions are added. 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 our algorithm. While values for these parameters cannot be randomly assigned, the automatic segmentation results are not very sensitive to them. One may refer to the guideline for assigning values to these parameters in section 3.

In this work, 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, our 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 [19].

We demonstrated an algorithm for humanoid robot dynamics optimization for motion regeneration. The optimal action selection strategy in Eq. (9) selects an action that minimizes the magnitude of the angular velocity of the robot's center of mass. This is a simple, implicit strategy to minimize the angular momentum of the whole-body motion of the robot. Minimizing angular momentum of the whole-body motion is a condition to achieve a faster dynamically stable gait. We took advantage of the single angular parameter of the motion in the low-dimensional space to create a compact predictive sensorimotor mapping model. Then, we used an optimization algorithm to search for an optimal motor command from the predictive model for each time step. By using a series of optimized motor commands, we successfully reproduced the motion. Note that even though just the walking straight motion is demonstrated in this paper, our motion regeneration algorithm can be applied to the other motions in the motion sequence, such as turning and side-stepping. In this work, the optimization algorithm only strictly searches on the curve that is constrained by CNLPCA. The optimization results do not yield any novel posture in the motion. The optimization algorithm only selects a more appropriate posture in the set of original postures subject to the objective function.

Thus, only the timing of the motion is optimized. For the optimization algorithm to be able to derive new postures, it may need the ability to search in all three dimensions [2] of the feature space instead of strictly searching on the one dimensional space of the single angular parameter. When a new posture is derived from three-dimensional optimization, we can achieve a more stable motion or lower gyroscope signal oscillation. For example, a high-speed walking gait may require a posture that has more lateral movement, which cannot be found in the original set of the postures. We plan to study

this three-dimensional optimization strategy.

REFERENCES

[1] Barbic, J., Safonova, A., Pan, J.Y., Faloutsos, C., Hodgins, J.K., Pollard, N.: Segmenting motion capture data into distinct behaviors. In: In Proceedings of Graphics Interface 2004, pp. 185 – 194 (2004)

[2] Chalodhorn, R., Grimes, D.B., Maganis, G.Y., Rao, R.P.N., Asada, M.: Learning humanoid motion dynamics through sensory-motor mapping. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA., pp. 3693 – 3698. IEEE Press, San Francisco, CA (2006)

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

[4] Ijspeert, A.J., Nakanishi, J., Schaal, S.: Trajectory formation for imitation with nonlinear dynamical systems. In: Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems, Hawaii, USA, pp. 752–757 (2001)

[5] Inamura, T., Toshima, I., Nakamura, Y.: Acquisition and embodiment of motion elements in closed mimesis loop. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA, pp. 1539–1544 (2002)

[6] Jenkins, O.C., Mataric, M.J.: Automated derivation of behavior vocabularies for autonomous humanoid motion. In: AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 225–232. ACM Press, New York, NY, USA (2003)

[7] Kirby, M.J., Miranda, R.: Circular nodes in neural networks. Neural Comput. **8**(2), 390–402 (1996)

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

[9] Lang, K.J., Waibel, A.H., Hinton, G.E.: A time-delay neural network architecture for isolated word recognition. Neural Netw. **3**(1), 23–43 (1990)

[10] MacDorman, K.F.: Feature learning, multiresolution analysis, and symbol grounding. In: Behavioral and Brain Sciences, vol. 21(1), pp. 32–33 (1998)

[11] MacDorman, K.F.: Grounding symbols through sensorimotor integration. Journal of the Robotics Society of Japan **17**(1), 20–24 (1999)

[12] Malthouse, E.: Limitations of nonlinear pca as performed with generic neural networks. Neural Networks, IEEE Transactions on **9**(1), 165–173 (Jan 1998). DOI 10.1109/72.655038

[13] Michel, O.: Webots: Symbiosis between virtual and real mobile robots. In: J.C. Heudin (ed.) VW '98: Proceedings of the First International Conference on Virtual Worlds, *Lecture Notes in Computer Science*, vol. 1434, pp. 254–263. Springer-Verlag, London, UK (1998)

[14] Okada, M., Tatani, K., Nakamura, Y.: Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA, pp. 1410–1415 (2002)

[15] Saul, L.K., Roweis, S.T.: Think globally, fit locally: Unsupervised learning of low dimensional manifold. Journal of Machine Learning Research **4**, 119–155 (2003)

[16] Smart, W.D., Kaelbling, L.P.: Practical reinforcement learning in continuous spaces. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Standord, CA, USA, June 29 - July 2, 2000, pp. 903–910 (2000)

[17] Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)

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

[19] Takahashi, Y., Hikita, K., Asada, M.: Incremental purposive behavior acquisition based on self-interpretation of instructions by coach. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, U.S.A., October 2003, vol. 1, pp. 686–693 (2003)

[20] Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)

[21] Zatsiorsky, V.M.: Kinematics of Human Motion, 1 edn. Human Kinetics Publishers; 1 edition, Champaign, IL (1997)