Learning to Imitate Human Actions through Eigenposes

Rawichote Chalodhorn and Rajesh P.N. Rao

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 the environment in order to devise complex control algorithms for generating a stable dynamic motion. Training using human motion capture (mocap) data is an intuitive and flexible approach to programming a robot but direct usage of kinematic data from mocap usually results in dynamically unstable motion. Furthermore, optimization using mocap data in the high-dimensional full-body joint-space of a humanoid is typically intractable. In this chapter, we purposes a new model-free approach to tractable imitation-based learning in humanoids by using *eigenposes*.

The proposed framework is depicted in Fig. 1. A motion capture system transforms the Cartesian positions of markers attached to the human body to joint angles based on kinematic relationships between the human and robot bodies. Then, linear PCA is used to create eigenpose data, which are representation of whole-body posture information in a compact low-dimensional subspace. Optimization of whole-body robot dynamics to match human motion is performed in the low dimensional subspace by using eigenposes. In particular, sensory feedback data are recorded from the robot during motion and a causal relationship between eigenpose actions and the expected sensory feedback is learned. This learned sensory-motor mapping allows humanoid motion dynamics to be optimized. An inverse mapping that maps optimized eigenpose data from the low-dimensional subspace back to the original joint space is then used to generate motion on the robot. We present several results

Rawichote Chalodhorn
Neural Systems Laboratory, Department of Computer Science and Engineering,
University of Washington, Seattle, WA 98195-2350 U.S.A.
e-mail: choppy@cs.washington.edu

Rajesh P.N. Rao
Neural Systems Laboratory, Department of Computer Science and Engineering,
University of Washington, Seattle, WA 98195-2350 U.S.A.
e-mail: rao@cs.washington.edu

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 357–381.
springerlink.com

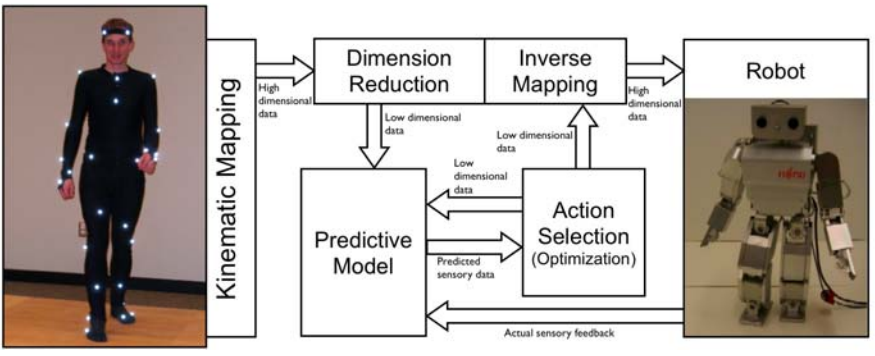

Fig. 1 A framework for learning human behavior by imitation through sensory-motor mapping in low dimensional subspaces.

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 physics-based model of the robot.

1 Related Work

Imitation is an important learning mechanism in many biological systems including humans [16]. 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) [18], imitation-based learning is fast: given a demonstration of a desired behavior, 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. Stable imitation requires deriving appropriate action commands that matches the robot's dynamics and the dynamic interaction between the robot and its environment. Sensory feedback data must also be taken into account for achieving stable imitation.

The idea of using imitation to train robots has been explored by a number of researchers. Demiris and Hayes [5] demonstrated imitative learning using a wheeled mobile robot that learned to solve a maze problem by imitating another homologous robot. Billard [2] 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. [8] designed a nonlinear dynamical system to imitate trajectories of joints and end-effectors of a human teacher; the robot learned and performed tennis swing motions by imitation. The mimesis theory of [9, 4] is based on action acquisition and action

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

Traditional model-based approaches based on zero-moment point (ZMP) [20, 17] or the inverted pendulum model [11, 10] 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. [15] demonstrated that stepping and walking policies could be improved by using RL and kernel dimension reduction (KDR): stepping and walking controllers are provided, and the learning system improves the performance of these controllers. The framework proposed in this chapter does not assume a specific type of nonlinear dynamical system or a specific gait as in [15] but is designed for learning general human motion from demonstrations. It can be used for learning different gaits for different tasks without redesign the algorithm.

Gaussian Process Dynamical Model (GPDM) [21] is a dimensionality reduction method for modeling high-dimensional sequential data. A temporal sequence of human walking data was modeled and reproduced without prior information. The resulted walking gait was reproduced kinematically without involving interactions with the environment. In contrast, the motion learning framework proposed in this chapter learns a dynamic model of interaction between the robot and its environment by learning a causal relationship between low-dimensional posture commands and sensory feedback.

2 3-D Eigenposes

Nonlinear dimensionality reduction algorithms had previously been applied to representation of human posture [3, 7]. Tatani and Nakamura [19] explored using a low-dimensional subspace to kinematically reproduce human motion on a humanoid robot via non-linear principal components analysis (NLPCA) [13]. However, these methods have some parameters that have to be well-tuned. Properties of the resulting low-dimensional subspaces used in these algorithms have not been well studied. Principal components analysis (PCA) is a linear dimensionality reduction technique whose properties have been well studied. We utilize PCA for the motion learning framework in this chapter.

2.1 Eigenposes as Low-Dimensional Representation of Postures

Particular classes of motion such as walking, sidestepping, or reaching for an object are intrinsically low-dimensional. We apply linear PCA to parameterize the low-dimensional motion subspace $\mathbb{X}$. Vectors in the high-dimensional joint angle space are mapped to the low-dimensional space by multiplication with the transformation matrix $\mathbf{C}$. The rows of $\mathbf{C}$ consist of the eigenvectors, computed via

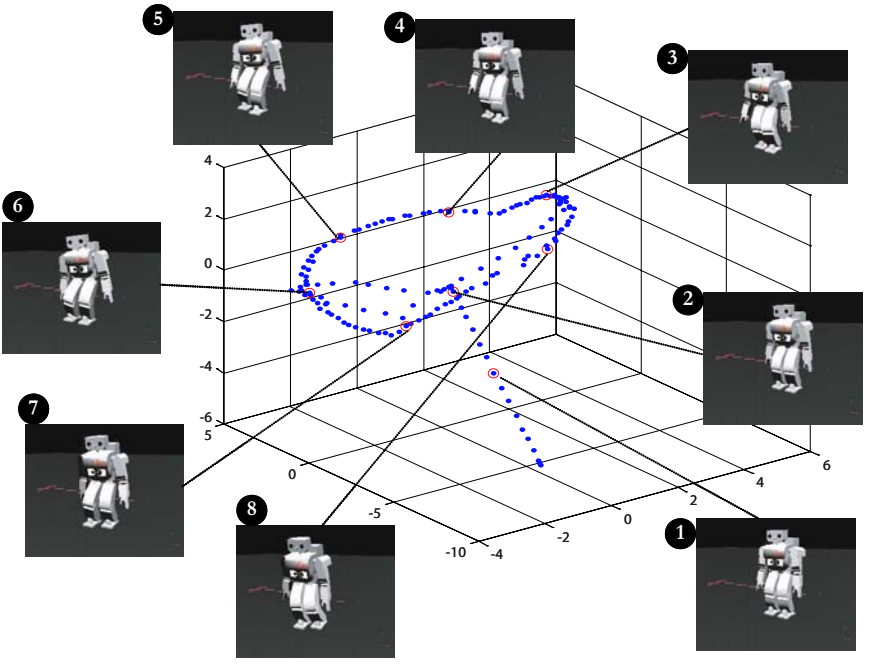

Fig. 2 Posture subspace and example poses from a hand coded walking gait. A three-dimensional space produced by PCA represents the posture of the Fujitsu HOAP2 robot. Blue points along a loop represent different robot postures during a single walk cycle. The first two labeled postures are intermediate postures between an initial stable standing pose and a point along the periodic gait loop represented by postures three through eight.

eigenvalue decomposition of the motion 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*.

For example, let Θ be the 20×1 vector of joint angles (the high-dimensional space) and $\mathbf{x}$ be the 3×1 vector of eigenpose in a 3D subspace. We can calculate $\mathbf{x}$ by first calculating $\mathbf{p} = \mathbf{C}\Theta$, where $\mathbf{p}$ is a 20×1 vector of all principal component coefficients of Θ and $\mathbf{C}$ is the 20×20 PCA transformation matrix. We then pick the first three elements of $\mathbf{p}$ (corresponding to the first three principal components) to be $\mathbf{x}$. The inverse mapping $\widetilde{\Theta}$, which is an approximation to Θ, can be computed by $\widetilde{\Theta} = \mathbf{C}^{\mathbf{T}}\tilde{\mathbf{p}}$ when the first three components of a full-rank-vector $\tilde{\mathbf{p}}$ are the elements of $\mathbf{x}$ and the rest of the elements are zero.

Examples of the low dimensional representation of the joint angle space of a HOAP-2 robot executing a walking gait (in the absence of gravity) are shown in Fig. 2. The robot images in the figure were produced by inverse PCA mapping. The figure demonstrates that the temporal sequence of motion data is still preserved in the low-dimensional subspace representation of the motion.

2.2 Action Subspace Embedding

PCA reduces the redundancy of posture data in high dimensional joint space. We use the reduced dimensional subspace $\mathbb{X}$ for constraining the postures of a motion pattern.

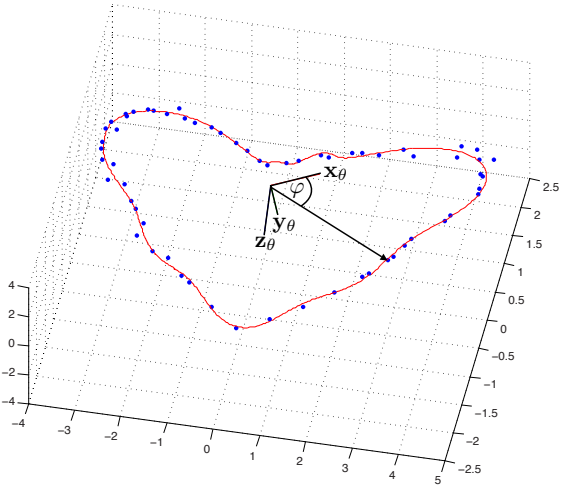

Fig. 3 Embedded action subspace of a humanoid walking gait. Training data points in the reduced posture space (shown in blue-dots) are converted to cylindrical coordinates relative to the coordinate frame $\mathbf{x}_\theta, \mathbf{y}_\theta, \mathbf{z}_\theta$. The points are then represented as a function of the phase angle φ, which forms an embedded action subspace (shown as a red solid-line curve).

A periodic movement such as walking can be represented by a closed-curve pattern $\mathbb{X}$. As an example, the periodic part of the data in Fig. 2 was manually segmented and is shown as the blue dots pattern in Fig. 3. 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. In particular, a one-dimensional representation of the original motion in the three-dimensional subspace is obtained by converting each point to its representation in a cylindrical coordinate frame. This is done by establishing a coordinate frame with three basis directions $\mathbf{x}_\theta, \mathbf{y}_\theta, \mathbf{z}_\theta$. 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 the resulting data is labeled $\hat{\mathbf{x}}^i$.

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

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

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:

$$[r, h] = g(\varphi) \tag{2}$$

where $0 \leq \varphi \leq 2\pi$. This function is approximated using a radial basis function (RBF) network. Note that the angle φ can be interpreted as the motion phase angle because it indicates how far the current posture is from the beginning of the motion cycle, which in our case is a walking gait. The first-order time derivative of φ tells us the speed of movement.

2.3 Action Subspace Scaling

The high-dimensional joint angle data are normalized before PCA. The data for each joint dimension are originally in different scales of values, but after normalization, they are scaled to the same range. When the normalized data are multiplied by a scalar value, the results are similar postures but with a different magnitude, allowing posture scaling. Note that posture scaling yields reasonable results only when the motion data set contains one specific type of motion.

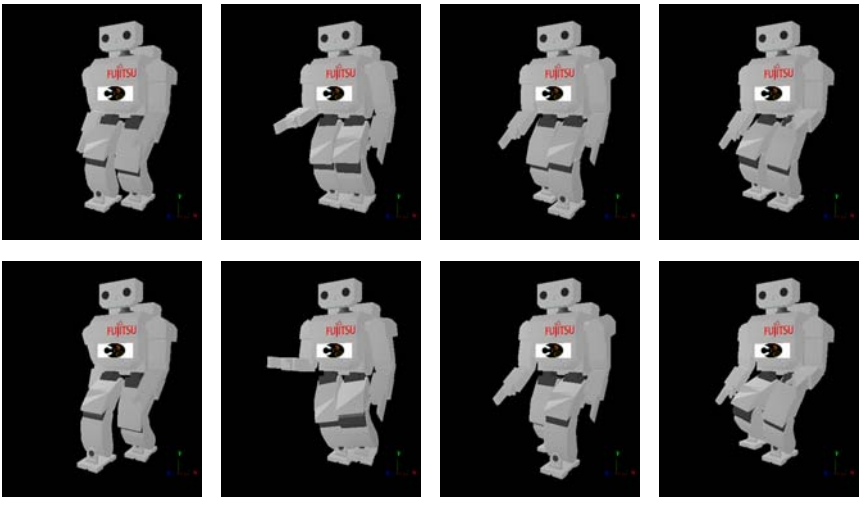

Fig. 4 Motion scaling of a walking gait. The first row shows four different postures of a walking gait. The second row shows coherent postures of the first row when a multiplying factor $f = 2.0$ is applied to the low-dimensional representation of this walking gait.

Scaling up and down motion patterns in the low-dimensional subspace produces similar motion patterns but with differences in the magnitude of motion. This means posture scaling ability is preserved after PCA is applied. If $\mathbf{A}$ represents a walking gait, multiplying $\mathbf{A}$ by a factor $f > 1$ will result in a similar walking gait but with larger steps. Multiplying $\mathbf{A}$ by a factor $f < 1$ results in a walking gait with a smaller step size. Fig. 4 shows an example of posture scaling of a walking motion.

It should be noted that action subspace scaling only produces similarity in 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 will be described in Section 5.1. We use action subspace scaling for $f < 1$ to initialize the learning process.

3 Learning to Predict Sensory Consequences of Actions

A key component of the proposed framework is learning to predict future sensory inputs based on current 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 $\mathbf{S} = \Theta \times \mathbf{P}$ is the Cartesian product of the high-dimensional joint space Θ and the space of other percepts $\mathbf{P}$. Other percepts include, for example, measurements from the torso accelerometer or gyroscope, foot pressure sensors, and information from camera images. The goal is to learn a function $F : \mathbf{S} \times \mathbf{A} \mapsto \mathbf{S}$ that maps the current state and action to the next state. In this framework, F is assumed to be deterministic.

Often the perceptual state s_t by itself 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:

$$s_{t+1} = F(s_t, s_{t-1}, ..., s_{t-n-1}, a_t, a_{t-1}, ..., a_{t-n-1}) \tag{3}$$

For this chapter, unless stated otherwise, we use a second-order $(n = 2)$ time-delay radial basis function (RBF) network for learning the predictive model. The state

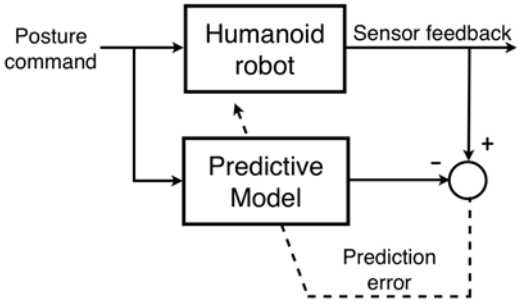

Fig. 5 Sensory signals (e.g., gyroscope signals) at the next time step are predicted based on the current posture command (eigenpose) as well as sensory signals and posture commands from previous time steps. The predictor is learned by comparing the predicted sensory signals to the actual sensor readings from the robot.

vector is taken to be 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 the learning method is shown in Fig. 5.

4 Predictive Model Motion Optimization

The algorithm presented in this section combines optimization and sensory prediction (see previous section) to select an optimal action plan for a humanoid robot. Fig. 6 illustrates this optimization process.

One may express the desired sensory states that the robot should attain during a particular class of actions using 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 $\Gamma(\mathbf{s})$:

$$a_t^* = \arg\min_{a_t} \Gamma(F(s_t, \ldots, s_{t-n-1}, a_t, \ldots, a_{t-n-1})). \tag{4}$$

In our work, the objective function Γ measures torso stability as defined by the following function of gyroscope signals:

$$\Gamma(\omega) = \lambda_x \omega_x^2 + \lambda_y \omega_y^2 + \lambda_z \omega_z^2, \tag{5}$$

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

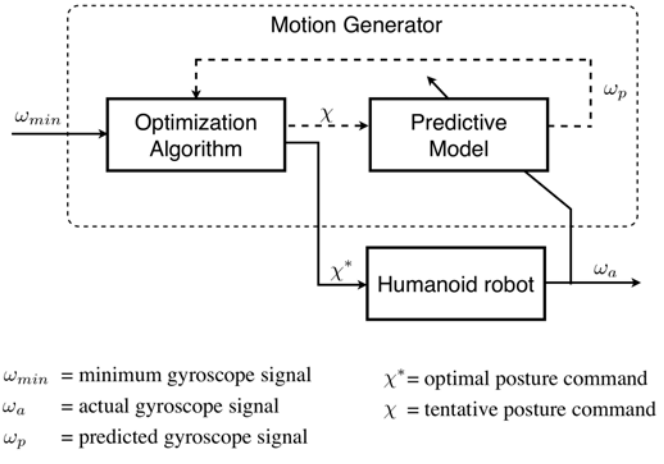

Fig. 6 Model predictive motion generator for optimizing motion stability. At time t, the optimization algorithm generates tentative actions or posture commands ($a_t \equiv \chi \in \mathbb{X}$). The predictive model predicts values of subsequent gyroscope signals ω_p. The optimization algorithm then selects the optimal posture command χ^* based on ω_p. The optimal posture command χ^* is executed by a robot/simulator. The resulting gyroscope signals are recorded for retraining the predictive model.

the starting posture is statically stable, one may simply minimize overall rotation of the robot body during motion in order to maintain balance by minimizing the sum of squares of the gyroscope signals. The objective function (5) thus 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:

$$\chi_t^* = \arg\min_{\chi_t \in S} \Gamma(F(\omega_t, \omega_{t-1}, \chi_t, \chi_{t-1})) \tag{6}$$

$$S = \left\{ \left[\varphi_s \ r_s \ h_s \right]^T | \varphi, r, h \text{ defined in Section 2.2} \right\} \tag{7}$$

For efficient optimization, the search space is restricted to a local region in the action subspace:

$$\varphi_{t-1} < \varphi_s \leq \varphi_{t-1} + \varepsilon_\varphi \tag{8}$$

$$r_a - \varepsilon_r \leq r_s \leq r_a + \varepsilon_r \tag{9}$$

$$h_a - \varepsilon_h \leq h_s \leq h_a + \varepsilon_h \tag{10}$$

$$0 < \varepsilon_\varphi < 2\pi \tag{11}$$

$$[r_a, h_a] = g(\varphi_s) \tag{12}$$

In the above, the phase-motion-command search-range φ_s begins after the position of the phase motion command φ_{t-1} at the previous time step, the range for the radius search r_s begins from a point in the action subspace embedding $\mathbf{A}$ defined by (12) in both positive and negative directions from r_a along $\mathbf{r}$ for the distance $\varepsilon_r \geq 0$, and the search range for h_s is defined in the same manner as r_s according to h_a and ε_h. In the experiments, the parameters $\varepsilon_\varphi, \varepsilon_r$,and ε_h were chosen to ensure efficiency while at the same time allowing a reasonable range for searching for stable postures.

Note that the search process exploits the availability of a human demonstrator by using the demonstrated action, as captured by (12), to constrain the search for stable robot actions. This imitation-based approach contrasts with more traditional approaches based on trial-and-error reinforcement learning or complex physics-based models.

Additionally, since selected actions will only truly be optimal if the sensory predictor is accurate, 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.

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

1. Use PCA to obtain eigenpose data from the human-demonstrated joint angle data.
2. Apply action subspace embedding 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 measurements and motor commands from the previous step to update the sensory-motor predictor as described in Section 3. In the present work, the state vector comprised of three channels of the gyroscope signal and the action variables were φ, r, and h in the low-dimensional subspace.
5. Use the learned model to estimate a sequence of actions according to the model predictive controller scheme described above (Fig.6).
6. Execute the computed actions and record sensory (gyroscope) feedback.
7. Repeat steps 4 through 6 until satisfactory motion is obtained.

4.1 One-Dimensional Motion Optimization

Our first experiment involved simulation in the Webots dynamic environment [14]. The goal was to increase the stability of a hand-coded walking gait (shown in Fig. 2) by using the motion optimization technique. The experiment also demonstrates the utility of action subspace embedding and the physical meaning of the parameter φ. Since this experiment involves one-dimensional optimization along φ, the parameters ε_r and ε_h in (9) and (10) are set to zero. Thus, (6) becomes:

$$\varphi_t^* = \arg\min_{\varphi_t} \Gamma(F(\omega_t, \omega_{t-1}, \varphi_t, \varphi_{t-1})). \tag{13}$$

We refer to this process as *motion-phase optimization* because only the parameter φ is optimized – the values of r and h are implicitly optimized through (2). The three channels of the gyroscope signal are regarded as *state* and the motion phase φ is regarded as the *action*. This *state-action* data is used for training the

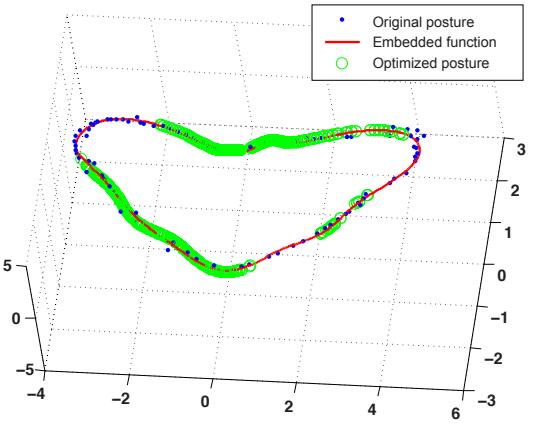

Fig. 7 Motion-phase optimization.

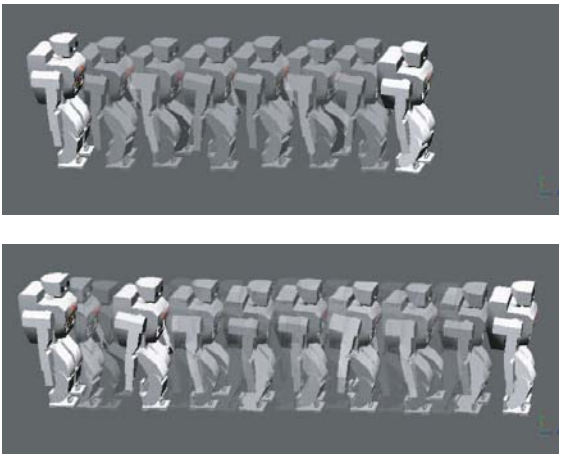

Fig. 8 Comparison of initial and optimized walking gait. The composite image at the top depicts the robot performing the initial walking gait for 2 seconds. Motion-phase optimization results in a faster walking gait as shown in the bottom image.

time-delay RBF network (depicted in Fig.5) to predict the gyroscope signals at the next time step. The optimization algorithm uses the learned predictor to obtain a new optimized action plan, which is then executed in the simulator.

The optimization result after three iterations of learning is shown in Fig.7. As seen in the figure, motion-phase optimization is essentially a line-search and 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 such a way that the selected actions minimize gyroscope signal oscillation. Fig. 8 shows that the optimized walking gait is significantly faster than the original gait. The walking speed of the optimized walking gait could be increased to three times the original gait in further optimization. Thus, we conclude that φ is controlling the timing of motion.

4.2 Three-Dimensional Motion Optimization

The second experiment focused on three-dimensional optimization of an initial walking gait based on equations (6)-(12). Since the optimization process is performed in the three-dimensional space of ϕ, $\mathbf{r}$ and $\mathbf{h}$ in cylindrical coordinates, novel postures resulting from optimized actions that do not lie on the constraint pattern can be expected.

The optimized walking gait in the low dimensional subspace shown in Fig. 9 was obtained after three iterations of sensory-motor prediction learning. An improved dynamically balanced walking gait was achieved. The new trajectory has a shape similar to the initial one but has a larger magnitude and is shifted. After remapping this trajectory back to the high dimensional space, the optimized motion pattern was

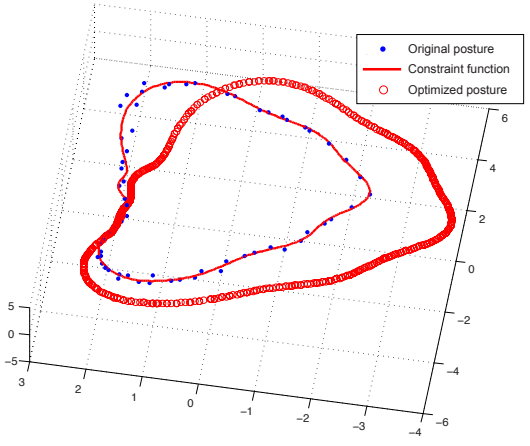

Fig. 9 Three-dimensional optimization results for a walking motion pattern based on an action subspace embedding in a low-dimensional subspace.

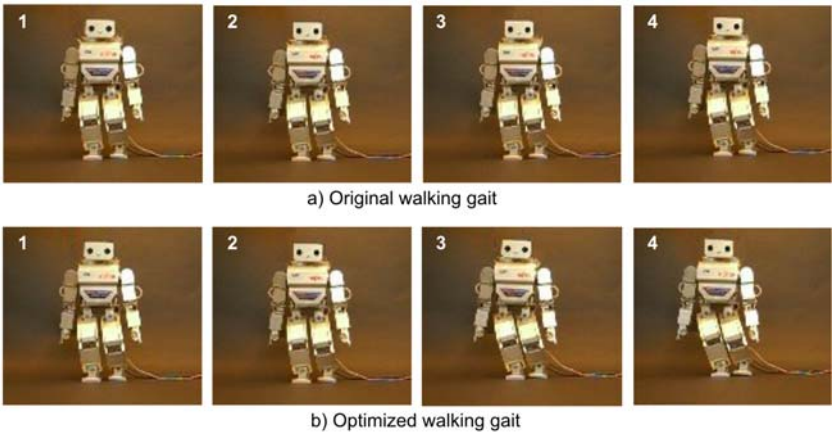

Fig. 10 Walking gait on a Fujitsu HOAP2 robot after three-dimensional optimization.

tested with the simulator and the real robot. The gyroscope readings from the new walking pattern are shown in Fig.11. The RMS values for the optimized walking gait along the **x**, **y** and **z** axes are 0.0521, 0.0501 and 0.0533 respectively, while the values for the original walking gait were 0.3236, 04509 and 0.3795. The RMS values for the optimized gait are thus significantly less than the original walking gait, indicating significant improvement in the dynamic stability of the robot. The robot walks with a larger step size but slower walking speed than the original walking gait.

The original and optimized gaits are shown in Fig. 10. The optimized walking gait has a different balance strategy compared to the original walking gait. In the

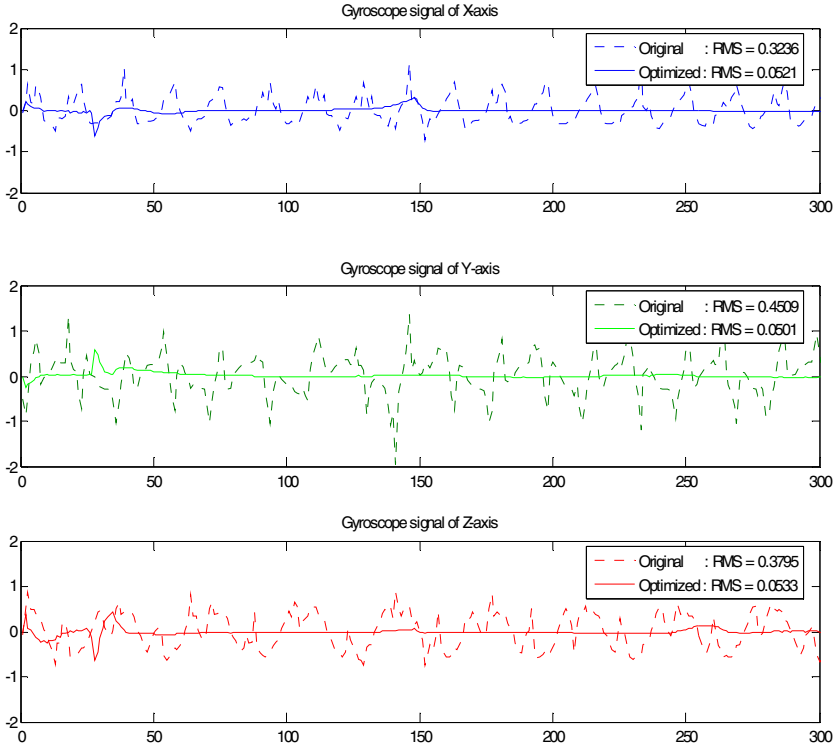

Fig. 11 Comparison of gyroscope signals before and after optimization. The plots from top to bottom show the gyroscope signals for the axes **x**, **y** and **z** recorded during the original and optimized walking motion. Notice that the root mean squared (RMS) values are significantly reduced for the optimized motion.

original gait, the robot quickly swings the whole body on the side of the support leg while it moves the other 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 other leg forward slowly. With the optimized gait, the robot also keeps its torso vertically straight throughout the motion. Fig.11 confirms that the algorithm was able to optimize the motion in such a way that the gyroscopic signals for the optimized motion are almost flat.

5 Learning Human Motion through Imitation

To be able to imitate a human motion, one must first solve the correspondence problem, which in our case is the problem of kinematic mapping of whole-body postures between a human demonstrator and a humanoid robot (we use a Fujitsu HOAP-2 robot). The human subject and the robot share similar humanoid appearances, but

their kinematic structure (skeletons) are dissimilar. The correspondence problem is solved by searching for a set of joint angles for the robot that generates the best matching pose with respect to the human demonstration. A Vicon optical system running at 120Hz and a set of 41 reflective markers was used for recording human motion. Initially, the markers are attached to the human subject and the 3-D positions of the markers are recorded for each pose during motion. The 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 used as positional constraints on the robot's skeleton and a set of joint angles is obtained using the standard numerical inverse kinematics (IK) solver in the Vicon motion capture system.

As depicted in Fig. 12, in order to generate robot joint angles, the human subject's skeleton is simply replaced by a robot skeleton of the same dimension. For example, the shoulders were replaced with three distinct 1-dimensional rotating joints rather than a single 3-dimensional human ball joint. The IK routine then directly generates the desired joint angles on the robot skeleton for each pose. One limitation of this technique is that there may be particular kinds of motion for which the robot's joints cannot approximate the human pose. This implies that the human demonstrator should try to avoid certain types of motion that the robot cannot imitate. For example, using toes in a walking gait should be avoided. In the case of arm movement, since the learner robot is a HOAP-2 robot having only four degrees of freedom (DoFs) in each arm, demonstration of actions that require six DoFs should be avoided. For the present work, since we only considered human motion that the robot has the potential to achieve, the above method proved to be a very efficient way of generating large sets of human motion data for robotic imitation.

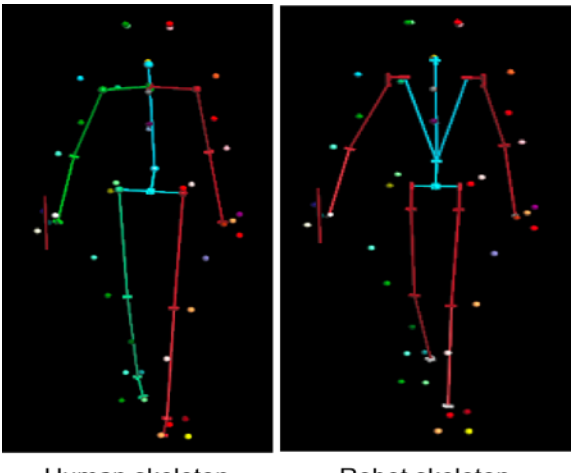

Fig. 12 Human skeleton (left) and robot skeleton (right) for kinematic mapping.

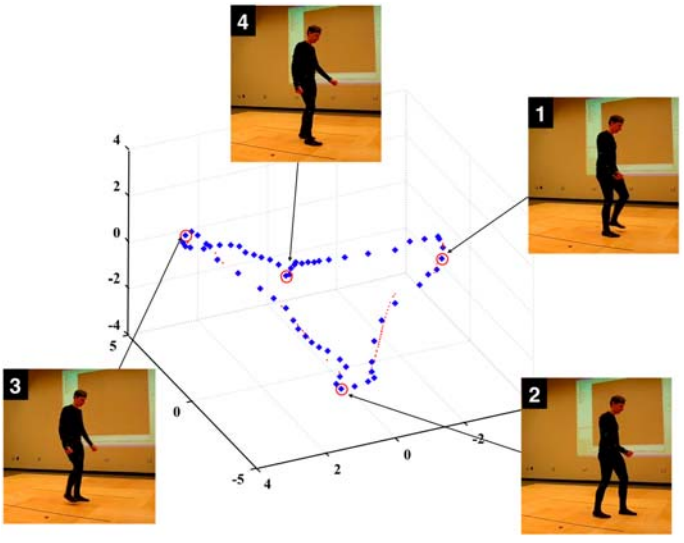

Fig. 13 Posture subspace and example poses obtained from human motion capture. Linear PCA was applied to joint angle data from a human kinematic configuration obtained via motion capture as described in Section 5. Blue diamonds represent different human postures during a single walking cycle. Red circles mark various example poses as shown in the numbered images.

5.1 Optimization of Motion Capture Data

Our first experiment with human motion capture data focused on making robot learn how to walk from a human demonstration of walking. An example of the low-dimensional representation of joint angle data from human walking are shown in Fig. 13. Note that the data pattern from human mocap data in Fig. 13 is more irregular than the data from the hand-coded walking gait in Fig. 2. Optimization based on human mocap data is difficult because the initial gait is unstable. We therefore used a motion scaling strategy in a low-dimensional subspace as described in Section 2.3. When the initial walking pattern in the low-dimensional subspace is scaled down, it produces smaller movements of the humanoid robot, resulting in smaller changes in dynamics during motion. The initial walking pattern is scaled down until a dynamically stable motion is found, after which the learning process is started. The motion optimization method in Section 4 is applied to the scaled-down pattern until its dynamic performance reaches an optimal level. The trajectory of the optimized result is gradually scaled up toward the target motion pattern. In this experiment, a scaling factor of 0.3 applied to the original motion pattern was found to be stable enough to start the learning process. The final optimization result is shown as a trajectory of red circles in Fig. 14. It corresponds to about 80% of the full scale motion from mocap data.

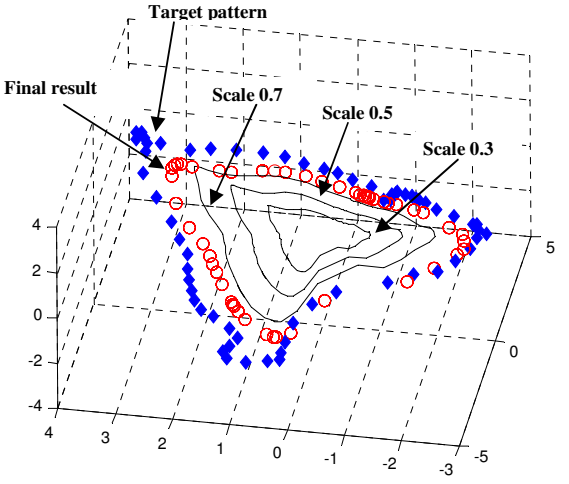

Fig. 14 Motion pattern scaling and optimization of human motion capture data. The target motion pattern is scaled down until it can produce a stable motion which is used to start the motion optimization process.

For the results in Fig. 14, five learning iterations with scale values 0.3, 0.5 and 0.7 were performed, and for the final result, ten iterations were performed for the scale 0.8. Note that the optimization time depends on the parameters $\varepsilon_\varphi, \varepsilon_r$ and ε_h. The parameter ε_φ must be defined such that value of φ_s is greater than the maximum difference in motion-phase-angle in the original mocap data. This will ensure that the optimization algorithm can search for a pose in a range that the original movement achieved. The longer the range of φ_s, the better the exploration but greater the optimization time. For r and h, the same parameter setup as φ can be applied. The value of ε_r and ε_h were set to 0.5 for all of the optimizations. The objective function in (5) has three tuning parameters, which are λ_x, λ_y and λ_z. At the beginning, we set the values of these parameters to 1. From observation of the first learning iteration, the parameters may be tuned, after which the values are maintained for the rest of learning iterations. In this chapter, λ_x and λ_z were set to 1.0. λ_y, which corresponds to the vertical direction, was set to 2.0 to allow the algorithm to compensate for the unexpected turns seen during the first learning iteration.

The simulation and experimental results are shown in Fig. 15. The learning process is performed in the simulator [14] and the resulting motion is tested on the real robot to minimize damage to the robot during the learning process. We observed that as expected, the walking gait on the real robot is not as stable as the results in the simulator because of differences in the frictional forces modeled in the simulator and the actual forces on the floor. We believe that performing further learning directly on the real robot (where permissible) could rectify this problem and improve performance.

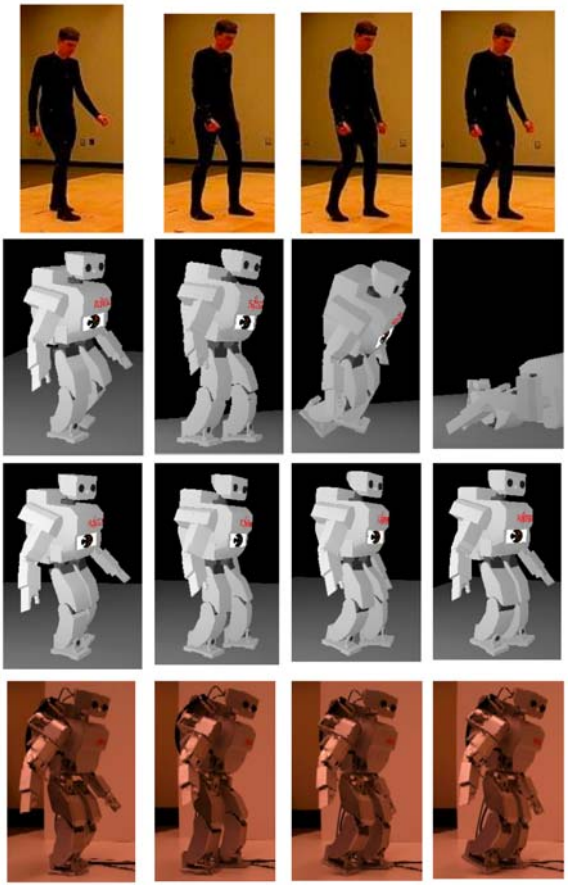

Fig. 15 Learning to walk through imitation. The first row shows a human subject demonstrating a walking gait in a motion capture system. The second row shows simulation results for this motion before optimization. The third row shows simulation results after optimization. The last row shows results obtained on the real robot.

Note that the learned motion is indeed dynamic and not quasi-static motion because there are only two postures in the walking gait that can be considered statically stable, namely, the two postures in the walking cycle where the two feet of the robot contact the ground. The rest of the postures in the walking gait do not need to be statically stable to maintain balance.

6 Lossless Motion Optimization

In the previous sections, the eigenposes that have been used for motion learning were three-dimensional. 3-D data are convenient for visualization and for

developing motion optimization algorithms based on analytical geometry. Periodic motion patterns such as hand coded walking and human mocap walking can be successfully learned using 3-D eigenposes but for some motion patterns, using only three dimensions cannot preserve the significant characteristics of the original motion. In this section, we extend our technique to larger-dimensional eigenposes. In particular, we show how the phase-motion optimization concept in Section 4.1 can be implemented with a new cylindrical coordinate transformation technique for large dimensional subspaces. We demonstrate how the algorithm can be used in a HOAP-2 humanoid robot to learn a sidestepping motion from a human demonstrator using a motion capture system.

6.1 Human Motion Capture of Sidestepping Motion

A motion capture session of a human demonstrator performing a sidestepping motion (to the right) as shown in Fig.16 was used as the target motion to be imitated. The motion sequence can be divided into four major steps starting from a standing posture. First, the right leg is lifted off the ground. Second, the right leg lands on the ground. Third, the left leg lifts off the ground. Fourth, the left leg swings in toward the right leg. For purposes of later discussion, we define the sidestep motion as being comprised of four phases: *right-lift, right-landing, left-lift*, and *left-landing*.

The kinematic mapping process described in Section 5 resulted in 20 dimensions of joint angle data, which were transformed into orthogonal principal axes using PCA as described in Section 2.1.

Fig.17 plots the accuracy of data reconstruction as a function of the number of principal components of the mocap data. When only the first three principal components are used, less than 80% of accuracy is achieved in reconstructing the original joint angle data. Accuracy increases gradually until 100% accuracy is obtained

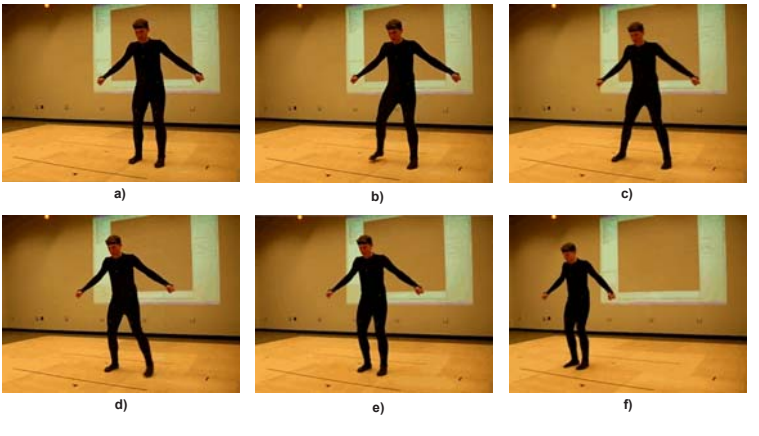

Fig. 16 Motion capture of sidestepping motion. Six samples of a rightward sidestepping motion sequence are shown in a) through f). Note that f) is a standing posture after one cycle of sidestepping. Each sidestepping cycle takes about 1 second.

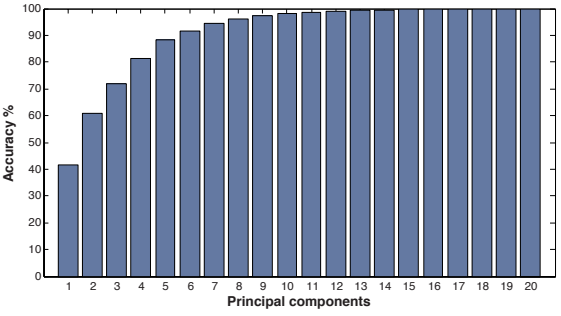

Fig. 17 Reconstruction accuracy as a function of the number of principal components of the sidestepping motion data from Figure 16.

when all 20 dimensions of eigenpose data are used. In this section, we illustrate our technique using 20-dimensional eigenposes for learning.

6.2 Large-Dimensional Cylindrical Coordinate System Transformation

The motion-phase optimization described above was 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, we suggest an extension of the cylindrical coordinate transformation idea to higher dimensions.

When $n = 3$, transformation from a Cartesian space $\mathbb{X}$ to a cylindrical coordinate system Φ is given by the mapping:

$$f(x,y,z) \rightarrow f(\varphi,r,h) \tag{14}$$

where

$$\varphi = \arctan(\frac{y}{x}), \tag{15}$$

$$r = \sqrt{x^2 + y^2}, \tag{16}$$

and

$$h = z. \tag{17}$$

When $n > 3$, an n-dimensional function may be written as:

$$f(d_1, d_2, d_3, \ldots, d_n) \tag{18}$$

where the d_i, $i = 1, \ldots, n$ $(n > 3)$, represent variables along orthogonal axes in $\mathbb{R}^n$.

We can express the function in (18) as:

$$f(x, y, z_1, \ldots, z_{n-2}) \tag{19}$$

The key idea is to represent the function f using a set of multiple cylindrical coordinate frames. Suppose, for example, that f is a 5-dimensional function. Then, f can be expressed in the form of (19) as:

$$f(x, y, z_1, z_2, z_3). \tag{20}$$

We use a piecewise mapping of f to cylindrical coordinates as follows:

$$\begin{array}{ll} f(x,y,z_1) & f(\varphi, r, h_1) \\ f(x,y,z_2) \Rightarrow & f(\varphi, r, h_2) \\ f(x,y,z_3) & f(\varphi, r, h_3) \end{array} \tag{21}$$

where φ and r are defined by Equations (15) and (16). Similarly, h_1, h_2 and h_3 are defined as in Equation (17).

Thus, the n-orthogonal dimensions of f are mapped to multiple cylindrical coordinate systems as:

$$f(x, y, z_1, \ldots, z_{n-2}) \rightarrow f(\varphi, r, h_1, \ldots, h_{n-2}). \tag{22}$$

For the 20-dimensional eigenpose data for sidestepping, 18 cylindrical coordinate frames are used by the above method.

6.3 Motion-Phase Optimization of Hyperdimensional Eigenposes

Trying to perform optimization on all of the orthogonal components of high-dimensional eigenposes may be intractactable, due to the curse of dimensionality problem. We therefore extend the one-dimemsional motion optimization idea from Section 4.1 to the higher dimensional case. For 3-D data, the action subspace embedding (described in section 2.2) is a single parameter function of motion-phase angle φ that produces values for the radius r and the height h of a periodic motion pattern in a cylindrical coordinate system. For n-dimensional eigenpose data, the action subspace embedding is a single parameter function of motion-phase angle φ that produces values for $r, h_1, h_2, \ldots, h_{n-2}$ of a periodic motion pattern. In particular, for the sidestepping motion pattern, the action subspace embedding is given by:

$$[r, h_1, h_2, \ldots, h_{18}] = g(\varphi). \tag{23}$$

The motion-phase optimization procedure in (13) can be directly applied to (23).

Fig. 18 shows the result of optimization (after five learning episodes) in the 3-D coordinate frame defined by the first three principal axes. From the figure, it can be seen that the optimized eigenposes are points on the original motion pattern, but distributed differently from the original pattern. This is because the motion-phase

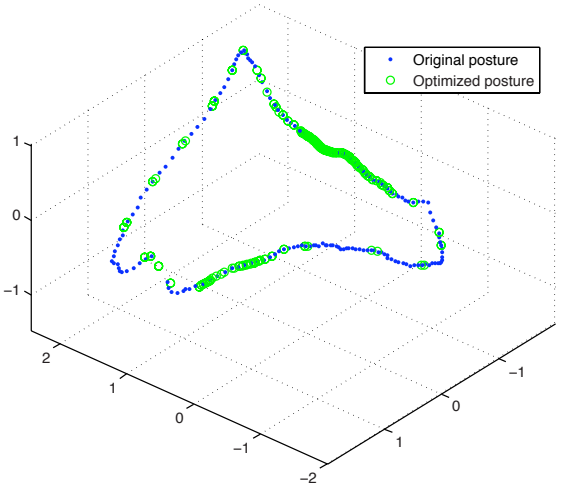

Fig. 18 Result of motion-phase angle optimization for sidestepping eigenpose data.

optimization is a one-dimensional optimization of the parameter φ in (23). The optimized eigenposes are thus strictly constrained to be within the original set of postures. The differences in distribution between the original pattern and the optimized pattern means that timing of postures during the motion have been altered. Notice in the figure that some gaps in the original pattern have been closed by the optimized postures. There are two reasons for this phenomena. First, the action subspace embedding is modeled as a closed-curve. Second, based on sensory feedback during learning episodes, the optimization algorithm found that it can achieve lower gyroscopic signal oscillation by choosing postures in the gap in the motion trajectory. As a result, the movement is smoother. Another attempt by the algorithm to obtain smoother movement can be noticed in the lower-left corner of the trajectory: the algorithm decided to plan the trajectory across the irregular corner of the original trajectory.

Fig. 19 show the simulation results for sidestepping motion. Column a) shows the original sidestepping motion 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. The first and the last rows are the standing postures at the beginning and end of the motion sequence, respectively. The second row is the right-lift phase. The third row is the right-landing phase. The fourth row is the left-lift phase, and the fifth row is the left-landing phase. In column b), the right foot of the robot was bouncing at the right-landing phase, causing the robot to be unable to lift its left foot up in the subsequent lift phase. As a result, the robot dragged its left foot along the ground during the left-landing period. This made the whole body of the robot turn, as can be observed in the last two rows of column b).

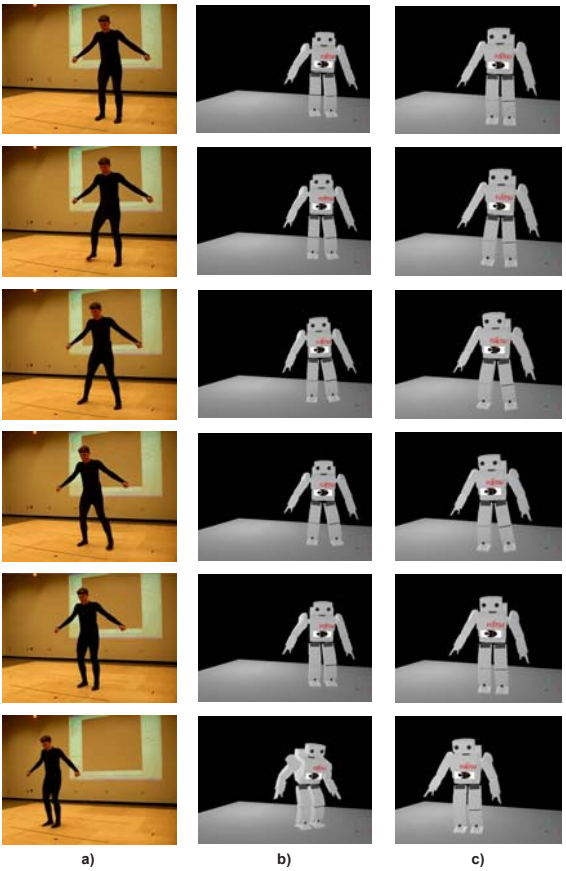

Fig. 19 Simulation results for sidestepping. Column a) shows original sidestepping motion sequence by the human demonstrator. Column b) shows the sidestep motion sequence on a HOAP-2 robot in a dynamics simulator without optimization at the motion scale 0.5. Column c) shows the sidestep motion after five learning episodes at motion scale 0.5.

In column c), the robot could perform the sidestep motion without the undesirable turn of the body.

While all of the key postures in column c) look very similar to the human postures in column a), timing of the movements are significantly different. The right-landing and left-landing phases of the optimized motion in column c) are relatively slower than the original human motion. These can also be observed in Fig. 18: there are two regions of the motion pattern with high density of optimized postures. These corespond to the slow landing phases. The slow landing phases also prevented the robot from dragging its left foot on the ground. As a result, the undesirable turn of the body was avoided and the sidestep motion was successfully learned.

7 Conclusion

This chapter proposed a framework that allows a humanoid robot to learn bipedal locomotion by imitation of human motion. Whole-body postures are represented using eigenposes computed from PCA. These eigenposes are used for learning a predictive sensory-motor model that allows a humanoid robot learn to walk by imitation of a human gait. Taken together, our results 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 coordinates not only further reduces dimensionality and complexity, but also provides meaningful variables in the low-dimensional subspace such as the *motion-phase-angle* φ and radius r. Optimization of the motion-phase-angle was shown to be equivalent to optimizing posture timing during the motion, while the radius r reflects magnitude of the motion, as determined by the first two principal components of the motion pattern. The physical meaning of the parameter h is yet to be clearly interpreted.

The human imitation-based learning framework described in this chapter demonstrates how a humanoid robot can learn basic human actions such as walking. These basic actions could be used a building blocks for learning more complex behaviors using approaches such as reinforcement learning. To learn actions other than walking and sidestepping, the objective function in (5) could be modified to accommodate different sensory variables. The robustness of the learned models to noise could be improved using a probabilistic approach as described in [6].

The proposed framework functions as an off-line motion planner rather than an on-line feedback controller. Thus, it cannot be applied directly to the problem of navigation on uneven terrain. One way of adding robustness to an off-line motion planner is to use a motion stabilizer [12], which is a combination of simple force/torque and gyroscope-based feedback controllers. We also investigating the possibility of a real-time feedback controller based on learning an inverse model of the predictor (3). Also under investigation are methods for learning non-periodic human motion as well as motion parameterization using eigenposes.

Acknowledgements. This work was supported by National Science Foundation (NSF) grant 0622252, the Office of Naval Research (ONR) Cognitive Science program, and a Packard Fellowship to RPNR.

References

1. Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, Taipei, Taiwan. IEEE, Los Alamitos (2003)
2. Billard, A.: Imitation: a means to enhance learning of a synthetic protolanguage in autonomous robots, pp. 281–310 (2002)

3. Chalodhorn, R., MacDorman, K.F., Asada, M.: An algorithm that recognizes and reproduces distinct types of humanoid motion based on periodically-constrained nonlinear pca. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 370–380. Springer, Heidelberg (2005)
4. Dana Kulic Dongheui Lee, C.O., Nakamura, Y.: Incremental learning of full body motion primitives for humanoid robots. In: IEEE International Conference on Humanoid Robots, pp. 326–332 (2008)
5. Demiris, J., Hayes, G.: A robot controller using learning by imitation. In: Proceedings of the 2nd International Symposium on Intelligent Robotic Systems, Grenoble, France (1994)
6. Grimes, D.B., Chalodhorn, R., Rao, R.P.N.: Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In: Sukhatme, G.S., Schaal, S., Burgard, W., Fox, D. (eds.) Robotics: Science and Systems. The MIT Press, Cambridge (2006)
7. Grochow, K., Martin, S.L., Hertzmann, A., Popovic, Z.: Style-based inverse kinematics. ACM Trans. Graph. 23(3), 522–531 (2004)
8. 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, pp. 752–757 (2001)
9. Inamura, T., Toshima, I., Nakamura, Y.: Acquisition and embodiment of motion elements in closed mimesis loop. In: ICRA, pp. 1539–1544. IEEE, Los Alamitos (2002)
10. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: ICRA [1], pp. 1620–1626
11. Kajita, S., Matsumoto, O., Saigo, M.: Real-time 3d walking pattern generation for a biped robot with telescopic legs. In: ICRA, pp. 2299–2306. IEEE, Los Alamitos (2001)
12. Kajita, S., Nagasaki, T., Kaneko, K., Yokoi, K., Tanie, K.: A running controller of humanoid biped hrp-2lr. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, Barcelona, Spain, April 18-22, pp. 616–622 (2005)
13. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. Journal of the American Institute of Chemical Engineers 37(2), 233–243 (1991)
14. Michel, O.: Webots: Symbiosis between virtual and real mobile robots. In: Heudin, J.-C. (ed.) VW 1998. LNCS (LNAI), vol. 1434, pp. 254–263. Springer, Heidelberg (1998)
15. Morimoto, J., Hyon, S.H., Atkeson, C.G., Cheng, G.: Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction. In: 2008 IEEE International Conference on Robotics and Automation, ICRA 2008, Pasadena, California, USA, May 19-23, pp. 2711–2716 (2008)
16. Rao, R.P.N., Shon, A.P., Meltzoff, A.N.: A Bayesian model of imitation in infants and robots. In: Nehaniv, C.L., Dautenhahn, K. (eds.) Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions. Cambridge University Press, UK (2007)
17. Sobotka, M., Wollherr, D., Buss, M.: A jacobian method for online modification of precalculated gait trajectories. In: Proceedings of the 6th International Conference on Climbing and Walking Robots, Catania, Italy, pp. 435–442 (2003)
18. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

19. Tatani, K., Nakamura, Y.: Dimensionality reduction and reproduction with hierarchical nlpca neural networks-extracting common space of multiple humanoid motion patterns. In: ICRA [1], pp. 1927–1932
20. Vukobratović, M., Yu, S.: On the stability of anthropomorphic systems. Mathematical Biosciences 15, 1–37 (1972)
21. Wang, J., Fleet, D., Hertzmann, A.: Gaussian process dynamical models. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems, vol. 18, pp. 1441–1448. MIT Press, Cambridge (2006)