Dana Kulić
Wataru Takano
Yoshihiko Nakamura
Department of Mechano-Informatics,
University of Tokyo,
7-3-1 Hongo, Bunkyo-ku,
Tokyo, Japan
{dana,takano,nakamura}@ynl.t.u-tokyo.ac.jp

Incremental Learning, Clustering and Hierarchy Formation of Whole Body Motion Patterns using Adaptive Hidden Markov Chains

Abstract

This paper describes a novel approach for autonomous and incremental learning of motion pattern primitives by observation of human motion. Human motion patterns are abstracted into a dynamic stochastic model, which can be used for both subsequent motion recognition and generation, analogous to the mirror neuron hypothesis in primates. The model size is adaptable based on the discrimination requirements in the associated region of the current knowledge base. A new algorithm for sequentially training the Markov chains is developed, to reduce the computation cost during model adaptation. As new motion patterns are observed, they are incrementally grouped together using hierarchical agglomerative clustering based on their relative distance in the model space. The clustering algorithm forms a tree structure, with specialized motions at the tree leaves, and generalized motions closer to the root. The generated tree structure will depend on the type of training data provided, so that the most specialized motions will be those for which the most training has been received. Tests with motion capture data for a variety of motion primitives demonstrate the efficacy of the algorithm.

KEY WORDS—learning and adaptive systems, cognitive robotics, gesture, posture, social spaces and facial expressions, human-centred and life-like robotics, humanoid robots, recognition, sensing and perception, computer vision

1. Introduction

In order for robots to operate successfully in human environments, they will need to be able to perform a wide variety of both precise and gross full body motions. In particular, in the case of humanoid robots, the ability to learn primitive and complex motion patterns by observing and imitating humans would be advantageous. Many algorithms in the literature have been developed for learning human motions through demonstration and imitation (Breazeal and Scassellati 2002; Schaal et al. 2003). However, most of these approaches consider the case where the number of actions to be learned are specified by the designer, the demonstrated actions are observed and clustered *a priori*, and the learning is a one-shot, off-line process. In this case, a global clustering method can be applied once all the data has been acquired, to determine the optimum number of motion primitives, and the allocation of the data to each primitive. However, a robot which is an inhabitant of the human environment should be capable of continuous learning over its entire lifespan. The robot should be able to observe, segment, and classify demonstrated actions on-line during co-location and interaction with the (human) teacher. During this type of learning, the number of motion primitives is not known in advance and may be continuously growing, and must be determined autonomously by the robot, as it is observing the motions. In addition, as the number of observed motions and learned motion primitives increases, the robot must be able to organize the acquired knowledge in an efficient and easily searchable manner.

Owing to the fact that motions are being segmented incrementally, before all the motions are known, the resulting segmentation will be analogous to a local optimization, and may be susceptible to local minima. In general, there will be a trade-off between classification accuracy and the number of training examples. As more examples become available, a more accurate clustering can be achieved. However, if the robot is able to extract appropriate motion primitives quickly, and after only a few examples are shown, the learned motion primitives can

The International Journal of Robotics Research
Vol. 27, No. 7, July 2008, pp. 761–784
DOI: 10.1177/0278364908091153

Figures 5, 11, 13, 14, 18, 22 appear in color online: http://ijr.sagepub.com immediately be used for motion generation. Once the robot can generate a learned motion, the learned motion can be further refined through other learning modalities, such as practice (Bentivegna et al. 2006) and feedback from the teacher (Nicolescu and Matarić 2005), which may be more effective than repeated observation alone. Therefore, we seek an algorithm which will accurately cluster with few examples. The ability to cluster quickly and accurately will depend on the similarity between the motions to be clustered, and the accuracy of the model used to represent each motion. For dissimilar motions, even a simple model can be successfully used, while for similar motions, a higher accuracy model is required to distinguish between motions. However, a higher accuracy model requires more memory and time resources. Therefore, it would be preferable if the model choice could be adaptable to the current level of similarity of the motions in the relevant region of the knowledge base.

Our motion model is inspired by the mirror neuron system, found in humans and other primates (Rizzolatti et al. 2001; Rizzolatti and Craighero 2004). The mirror neuron system is believed to be a direct-matching mechanism, whereby observed actions are understood when the visual representation of the observed action is mapped onto the motor representation of the same action. A motion is understood by causing the motor system of the observer to "resonate". The neurons which respond in this manner have been named *mirror neurons*. In previous research (Ezaki 2000; Inamura et al. 2004; Takano 2006), humanoid motion primitives have been encoded using hidden Markov models (HMMs), and subsequently used for motion generation. However, the initial training of the models was carried out off-line, where all the training examples for each model were grouped manually. In this paper, we develop an algorithm for incremental and autonomous acquisition and learning of the motion primitives, and the automated organization of the acquired behaviors.

In order to extract motion primitives during on-line observation, several key issues must be addressed by the learning system: automated motion segmentation, recognition of previously learned motions, automatic clustering and learning of new motions, and the organization of the learned data into a storage system which allows for easy data retrieval. In this paper, our focus is on the last three items: motion learning and recognition, clustering, and organization.

1.1. Related Work

Robot skill learning from observation is a long-standing area of research (Kuniyoshi et al. 1989; Kuniyoshi and Inoue 1994; Liu and Asada 1992; Dillmann et al. 1999). Breazeal and Scassellati (2002) and Schaal et al. (2003) provide an overview on recent research on motion learning by imitation. As noted by Breazeal and Scasellati, the majority of algorithms discussed in the literature assume that the motions to be learned are segmented *a priori*, and that the model training takes place off-line. Several different techniques have been applied to the modeling the motion primitives, including dynamical models (Nakanishi et al. 2004), neural networks (Ogata et al. 2005), and stochastic models (Inamura et al. 2004; Billard et al. 2006; Taylor et al. 2006).

Ogata et al. (2005) develop a connectionist architecture suitable for long-term, incremental learning. In their work, a neural network (NN) is used to learn a navigation task during cooperative task execution with a human partner. The authors introduce a new training method for the recursive neural network (RNN), which avoids the problem of memory corruption during new training data acquisition. Their approach is based on neuroscience research, which has shown that the hippocampus is used as temporary (short-term) memory, and that this memory is consolidated into long-term memory during sleep in a process analogous to rehearsal. In their architecture, previous training data is used together with new data during retraining, to avoid damaging previously learned knowledge. It is found that the consolidation RNN performs considerably better than the regular NN architecture, and continues improving over time. The consolidation RNN is also rated more favorably by subjects. However, in this case, the robot learns only one task, and no hierarchical organization of knowledge takes place.

Taylor et al. (2006) describe an approach for modeling human motion using a conditional restricted Boltzmann machine (CRBM). This stochastic model consists of a distributed hidden state, increasing the representation power of the model; however, unlike in a HMM, each hidden state variable is conditioned on the previous visible observation(s), rather than on the previous hidden state, and autoregressive connections are added between visible observation time steps. An efficient inference algorithm (maximum likelihood learning) is available for the CRBM, based on contrastive divergence. The learned model can generate continuous motion sequences, as well as learn the transitions between motions. Once the low-level model consisting of individual motion patterns has been trained, additional higher order layers can be added to model the higher order structure of motion patterns. However, this implies that all primitive level motions must be known and input into the model, so that the algorithm is not able to build the model incrementally, during on-line observation.

HMMs have been a popular technique for human motion modeling, and have been used in a variety of applications, including skill transfer (Yang et al. 1997; Dillmann et al. 1999), robot-assisted surgery (Kragic et al. 2005; Ekvall et al. 2006), sign language and gesture modeling (Startner and Pentland 1995; Bernardin et al. 2005; Iba et al. 2005), and motion prediction (Bennewitz et al. 2005; Ho et al. 2005). A common paradigm is *Programming by Demonstration* (PbD) (Dillmann et al. 1999; Dillmann 2004; Bernardin et al. 2005). While PbD is a general paradigm, in many of the systems demonstrated thus far, the number of motions is specified *a priori*, the motions are clustered and trained off-line, and then a static model is used during the recognition phase.

Calinon et al. (2007) describe a system for programming by demonstration. In their approach, the data (including both joint angle data and absolute and relative object position and hand position data) is first analyzed via principal components analysis (PCA) to determine the relevant subspace for the task (named the latent space). The reduced dimensionality data is then temporally aligned using dynamic time warping (DTW), which is a type of non-linear scaling which minimizes the error between the signals and a reference signal. Once the data has been temporally aligned, it is abstracted into a set of Gaussian mixtures. The number of mixtures is selected via the Bayesian information criterion (BIC). Once the number of mixtures is known, the expectation-maximization (EM) algorithm is used to find the Gaussian mixture model (GMM) parameters (the prior, mean, and covariance matrix). The GMM also includes the time parameter as an additional output variable. To evaluate the success of the imitation, an evaluation metric is defined in the reduced dimensionality subspace (the latent space), as a weighted square error between the candidate position and the desired position (the desired position is generated by the GMM model). By defining the cost function as a quadratic function, the problem can be reduced to finding an optimum subject to kinematic and environmental constraints. This system is also extended to an on-line, interactive approach by developing a method for incremental training of the GMM structure (Calinon and Billard 2007b) and developing an interactive training approach combining demonstration and kinesthetic training (Calinon and Billard 2007a).

Bennewitz et al. (2005) describe a system for estimating the motion patterns of humans in their environments. These patterns are then used to predict future motion and enable a mobile robot to plan trajectories to evade humans and successfully navigate in the human environment. In the proposed system, a set of trajectories between resting places (desks, TV, etc.) is used as the input data. The number of unique trajectories is determined automatically via the EM algorithm. The clustered trajectories are used to train a set of HMMs which abstract typical motion patterns. The models are then used to maintain a belief about the positions and future positions of persons, and enable the mobile robot to appropriately navigate given these predictions.

Ohkawa and Nakamura (2005) develop an off-line method for automated segmentation of motion patterns. The motion patterns are encoded into HMMs, and a hierarchical tree structure is built representing the relationship between the HMMs based on the intra-HMM Mahalanobis distance. However, this method can only be used once the (finite) training data set has been fully acquired.

Kadone and Nakamura (2005, 2006) describe a system for automated segmentation, memorization, recognition, and abstraction of human motions based on associative NNs with non-monotonic sigmoid functions. Their approach achieves on-line, incremental learning of human motion primitives, and self-organization of the acquired knowledge into a hierarchical tree structure. However, the abstracted motion representation can only be used for subsequent motion recognition, and cannot be used for motion generation.

Takano and Nakamura (2006) develop a system for automated segmentation, recognition, and generation of human motions based on HMMs. In their approach, a set of HMMs is trained incrementally, based on automatically segmented data. Each new motion sequence is added to the HMM which has the highest likelihood of generating the motion. The selected HMM is then trained for competitive learning with the new data. The number of HMMs is fixed and determines the level of abstraction. A choice of smaller number results in a higher level of abstraction. However, no mechanism is proposed for the emergence of a hierarchy among different levels of abstraction.

Another important issue during model learning is the selection of the model size and structure. Dixon et al. (2004) describe a system for partially automating the programming of an industrial robot manipulator by facilitating the definition of trajectory waypoints. Based on previous programming examples, the proposed system predicts where the user may move the end-effector and automatically moves the robot to that point. Continuous density HMMs are used to model previously observed user programming inputs. Relative position and orientation of the end effector are used as the observed data. Both the structure and the parameters of the HMM are trained on-line. Initially, each example is constructed as a front-to-back model, assigning each waypoint to one state. Next, similar nodes are iteratively merged, resulting in the final topology. To facilitate real-time operation, a one-shot training procedure is used to estimate the model parameters.

Ekvall et al. (2006) describe a system for robotic assistance during microsurgery, where adaptive virtual fixtures are used to guide the surgeon's hands. For each task, an HMM is trained by using support vector machines (SVM) to determine the appropriate number of states (corresponding to straight lines in the trajectory) and to estimate the probability density function for each state of the HMM. The HMM is then used to recognize the current state (line direction), and apply the appropriate fixture to constrain the manipulator's motion.

Billard et al. (2006) use HMM models for motion recognition and generation of humanoid motions. The BIC is used to select the optimal number of states for the HMM. In the experiments, three different tasks are demonstrated using the HOAP-2 humanoid robot. The robot is trained using kinesthetic training. However, all the exemplar motion patterns are acquired and grouped before the training begins, and the number of motions to be learned is specified *a priori*.

The use of the Akaike information criterion (AIC) (Kulic et al. 2007a) has also been proposed; however, both the AIC and BIC are based on a tradeoff between model performance at characterizing the observations, and the number of parameters, and both require a time-consuming search of the model space to find the best matching model. In the motion recognition domain, the model size required depends not only on the model performance for the current observation, but also on the structure of the knowledge database itself. If there are many similar motions in the database, a more accurate model is required, so that motions can be more easily discriminated. On the other hand, if motions are dissimilar, a very simple model can easily discriminate between them.

In this paper, a variable structure HMM-based representation is used to abstract motion patterns as they are perceived. Individual motion patterns are then clustered in an incremental fashion, based on intra-model distances. The resulting clusters are then used to form a group model, which can be used for motion generation. The model size is adjusted automatically on-line, based on the accuracy requirements in the given region of the knowledge space. A new algorithm for sequentially training the stochastic models is introduced, to allow fast, on-line training, and take advantage of existing knowledge stored in the lower-accuracy model. Section 2 describes the stochastic encoding of each motion primitive and model training algorithm. The incremental behavior learning and hierarchy formation algorithm is described in Section 3. Experimental results on a motion capture data set are described in Section 4; concluding remarks are presented in Section 5.

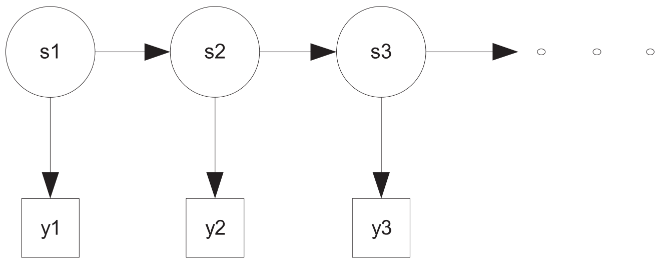

Fig. 1. The hidden Markov model.

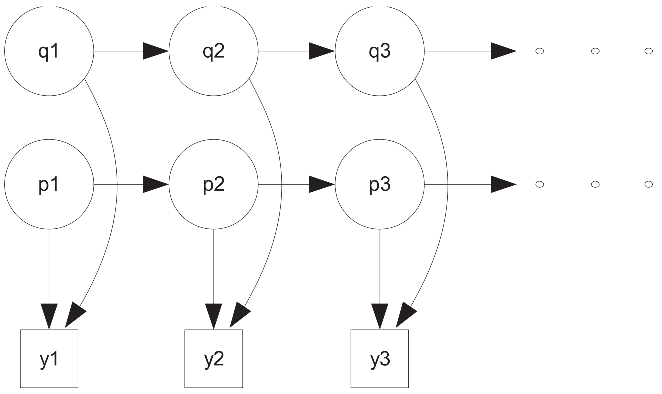

Fig. 2. The factorial hidden Markov model.

2. Factorial Hidden Markov Models

A HMM abstracts the modeled data as a stochastic dynamic process. The dynamics of the process are modeled by a hidden discrete state variable, which varies according to a stochastic state transition model $A[N, N]$, where N is the number of states in the model. Each state value is associated with a continuous output distribution model $B[N, K]$, where K is the number of outputs. Typically, for continuous data, a Gaussian or a mixture of Gaussian output observation models is used. HMMs are commonly used for encoding and abstracting noisy time series data, such as speech (Rabiner 1989) and human motion patterns (Billard et al. 2006; Takano 2006). Efficient algorithms have been developed for model training (the Baum–Welch algorithm), pattern recognition (the forward algorithm), and hidden state sequence estimation (the Viterbi algorithm) (Rabiner 1989). The Baum–Welch algorithm is a type of iterative EM algorithm. In the Expectation step (E-step), given a set of model parameters and a data sequence, the posterior probabilities over the hidden states are calculated. Then, in the Maximization step (M-step), a new set of model parameters are calculated which maximize the log likelihood of the observations.

Once trained, HMM can also be used to generate a representative output sequence by sampling the state transition model to generate a state sequence, and then sampling the output distribution model of the current state at each time step to generate the output time series sequence. A schematic of an HMM is shown in Figure 1.

A factorial hidden Markov model (FHMM) (Ghahramani and Jordan 1997) is a generalization of the HMM model, where there may be multiple dynamic processes interacting to generate a single output. In an FHMM, multiple independent dynamic chains contribute to the observed output. Each dynamic chain m is represented by its own state transition model $A_m[N_m, N_m]$ and output model $B_m[N_m, K]$, where M is the number of dynamic chains, N_m is the number of states in dynamic chain m, and K is the number of outputs. At each time step, the outputs from all the dynamic chains are summed, and output through an expectation function to produce the observed output. The expectation function is a multivariate Gaussian function with the chain output as the means, and a covariance matrix representing the signal noise. For example, FHMMs have been used to model speech signals from multiple speakers (Betkowska et al. 2006), and the dynamics of a robot and changing environment for simultaneous localization and mapping (Murphy 1999). Figure 2 shows a schematic of an FHMM with two dynamic chains.

An FHMM can be trained by an adaptation of the Baum–Welch algorithm (Ghahramani and Jordan 1997). However, this (exact) algorithm has a time complexity of $O(TMN^{M+1})$ where T is the length of the data sequence. This means that

```
1: procedure FHMMTrain
2:     Initialize $A_m[N_m, N_m]$, $B_m[N_m, K]$, $\pi_m(i)$ and $\Sigma$
3:     for $cycle \leftarrow 1, maxIterations$ do
4:         E-Step
5:         for $n \leftarrow 1, numSequences$ do
6:             Forward Algorithm
7:             Backward Algorithm
8:             Estimate $\gamma_t(i)$
9:             Estimate $\xi_t(i,j)$
10:            Estimate $\eta_t(m,n)$
11:        end for
12:        M-Step
13:            Calculate $B_m[N_m, K]$
14:            Calculate $\Sigma$
15:            Calculate $A_m[N_m, N_m]$
16:            Calculate $\pi_m(i)$
17:    end for
18: end procedure
```

Fig. 3. Generic FHMM training algorithm pseudocode.

the time complexity increases exponentially with the number of chains. This is due to the fact that, even though the dynamic chains are independent of each other, they become dependent given an observation sequence. Therefore, the E-step of the EM algorithm becomes intractable for a large number of chains. Faster, approximate algorithms, for which the time complexity is quadratic as a function of the number of chains, have been developed for FHMM training, which implement an approximate rather than the exact E-step. These are based on variational methods (Ghahramani and Jordan 1997), or generalized backfitting (Jacobs et al. 2002). During the E-step, the expectation of the vector state occupation probabilities $\gamma_t(i)$, the expectation of the matrix of state occupation probabilities at two consecutive time steps $\xi_t(i, j)$, and the expectation of the joint probability between the state vectors of two chains $\eta_t(m, n)$ are estimated. During the M-step, the covariance matrix Σ, the output model $B_m[N_m, K]$, the state transition model $A_m[N_m, N_m]$, and the initial state distribution probabilities $\pi_m(i)$ are updated. The pseudocode for a generic FHMM training algorithm as developed by Ghahramani and Jordan (1997) is summarized in Figure 3.

Once the FHMM is trained, pattern recognition can be implemented by an adaptation of the forward–backward algorithm (Ghahramani and Jordan 1997).

To generate a representative sequence from an FHMM, at each time step, the value of the hidden state for each chain is determined using the state transition model A_m, and the output contribution for each chain determined based on the output model for the current hidden state. The output is then formed by sampling from an output distribution whose expectation is the sum of all the respective chain outputs.

Due to the fact that the chains in an FHMM are evolving independently of each other during generation, this approach may not be suitable for generating continuous walking motions, but rather only walking segments. Continuous walking motion could be generated by inducing a dependence between the chains, by conditioning the generation process on an observation sequence most similar to the desired motion via the Viterbi algorithm, as described by Lee et al. (2008).

2.1. Sequential Training

FHMMs require a modification to the very efficient Baum–Welch training algorithm. Using either the Gharmani and Jordan approximate training algorithm (Ghahramani and Jordan 1997), or the generalized-backfitting algorithm (Jacobs et al. 2002), the chains are trained simultaneously, which significantly increases training time. Similarly, due to the more complex structure of the FHMM model, the recognition algorithm (the forward procedure) is more complex and time consuming as compared to a single-chain HMM. Therefore, we would like to use a compact representation (a single HMM chain) when patterns are easy to distinguish, and a more detailed representation (multiple chains) during generation and when motions are very similar to each other. For this approach, a modified training algorithm is developed, training the chains sequentially instead of simultaneously.

The developed approach is equivalent to a single pass of the generalized-backfitting algorithm (Jacobs et al. 2002), or the structured variational inference approximation (Ghahramani and Jordan 1997). A key difference between the proposed approach and a single pass of the generic generalized-backfitting algorithm is the re-weighting of the relative contributions of the chains, such that each chain contributes approximately equally to the observed output. Analyzing the parallel-trained FHMMs (Kulic et al. 2007b), it appears that FHMM chains trained parallely each approximate a (scaled) version of the motion data. To approximate this result with the sequential training algorithm, first, a standard HMM chain is trained on the motion data, using the Baum–Welch training algorithm. This chain alone can then be used for the initial recognition (during the tree search). Next, additional chains are trained on the error between the true data and the motion generated by the scaled sum of the preceding chains. The training data for each subsequent chain is calculated based on the observation data and the existing contribution from the previously trained chains.

$$e_t^n = \frac{1}{W}\left(y_t^n - \sum_{i=0}^{m-1} W C_i\right), \tag{1}$$

where e_t^n is the residual error (a set of N time series sequences), y_t^n is the original data (a set of N time series sequences), $W = 1/M$ is the weight applied to each chain, M is the new number of chains, and C_i is the contribution of each previously trained chain i.

The contributions from the previously trained chains can be calculated based on:

(a) the $\gamma_t(i)$ values for each chain, that is,

$$C_i = \sum_{j=0}^{J} \boldsymbol{\mu}_j^{(i)} \gamma_t^{(i)}(j), \tag{2}$$

where $\boldsymbol{\mu}_j^{(i)}$ is the (already trained) mean values vector for chain i and state j, and $\gamma_t^{(i)}(j)$ is the probability that state j in chain i is active at sample time t;

(b) the Viterbi sequence for each chain, that is,

$$C_i = \boldsymbol{\mu}_{j*}^{(i)}, \tag{3}$$

where $\boldsymbol{\mu}_{j*}^{(i)}$ is the mean value vector at the current state $j*$, as determined by the Viterbi algorithm for each time series sequence; or

(c) the generated sequence for each chain,

$$C_i = \hat{y}_t^i, \tag{4}$$

where $\hat{y}_t^i$ is the output vector generated by chain i at time step t.

Once the training data for the new chain are generated, the new chain is trained with the Baum–Welch algorithm. Following training, for the forward (recognition algorithm), the variance at each state combination is computed as follows:

$$\Sigma = \sum_{i=0}^{M} W^2 \Sigma_j^{(i)}, \tag{5}$$

where Σ is the resulting covariance, and $\Sigma_j^{(i)}$ is the covariance at the corresponding state j of chain i.

The pseudocode for the algorithm is outlined in Figure 4.

The developed algorithm approximates the actual FHMM distribution with a tractable distribution assuming that the Markov chains are independent given the data. In addition, the sequential approach allows the on-line segmenting algorithm to re-use the training information of the simpler model (i.e. the previously trained chains).

2.2. Human Motion Pattern Representation using FHMMs

HMMs have been widely used to model human motion data for both recognition and motion generation (Billard et al. 2006; Takano 2006). However, when using the same HMM structure for both recognition and generation, there is an inherent tradeoff when selecting the HMM model (i.e. the number of model states). An HMM model with a low number of states will be better at generalizing across variable data, and better at correctly recognizing new data. In general, an HMM with a low number of states (5–20) provides excellent recognition performance (Billard et al. 2006; Takano 2006). When automatically selecting the number of states based on BIC (Billard et al. 2006) or AIC (Kulic et al. 2007a), under 10 states are usually selected for typical human motion patterns such as walking, kicking, punching, etc. However, in this case, the likelihood of the model representing the data distribution is used as the criterion for selecting the appropriate number of states. At the same time, a model with a low number of states will not be able to faithfully reproduce the observed motion through generation. On the other hand, a large state model will be better at reproducing the observed motion, but will be prone to over-fitting. FHMMs, which use a distributed rather than a single multinomial state representation, provide a more efficient approach for combining good generalization for recognition purposes with sufficient detail for better generation. An alternate solution, proposed by Calinon and colleagues (Calinon and Billard 2007b,a; Calinon et al. 2007) is to use GMMs together with regression for generating motions.

3. Incremental Behavior Learning and Hierarchy Formation

Each time a new motion sequence is observed, the robot must decide if the observed motion is a known motion, or a new motion to be learned. In addition, over the lifetime of the robot, as the number of observed motions becomes large, the robot must have an effective way of storing the acquired knowledge for easy retrieval and organization. In the proposed approach, a hierarchical tree structure is incrementally formed representing the motions learned by the robot. Each node in the tree represents a motion primitive, which can be used to recognize a similar motion, and also to generate the corresponding motion for the robot. Rather than using a fixed size model for the motions, the model accuracy is adjusted based on the recognition requirements in the given region of the knowledge database. Initially, each motion and motion group is encoded as a simple HMM, with few states. As the required model complexity increases, additional dynamic chains are added to the model, to form FHMMs (Ghahramani and Jordan 1997) (see Figure 2).

An overview of the algorithm and the incremental hierarchy formation is shown in Figure 5. The algorithm initially begins with one behavior group (the root node). Each time a motion is observed from the teacher, it is encoded into a HMM (Figure 5(a)). The encoded motion is then compared to existing behavior groups via a tree search algorithm, using the symmetric model distance measure (Rabiner 1989; Kulic et al. 2007b) (Figure 5(b)), and placed into the closest group (Figure 5(c)). Each time a group is modified, a hierarchical agglomerative clustering algorithm (Jain et al. 1999) is performed within the exemplars of the group (Figure 5(d)). If a

```
1: procedure SequentialFHMMTrain
2:     Initialize first chain
3:     $A_0[N_0, N_0]$, $B_0[N_0, K]$, $\pi_0(i)$
4:     Baum–Welch algorithm: train chain on time series $y_t^n$
5:     for $m \leftarrow 1, M$ do
6:         Initialize next chain
7:         $A_m[N_m, N_m]$, $B_m[N_m, K]$, $\pi_m(i)$
8:         Calculate residual error $e_t^n = \frac{1}{W}(y_t^n - \sum_{i=0}^{m-1} WC_i)$
9:         Baum–Welch algorithm: train chain on time series $e_t^n$
10:    end for
11: end procedure
```

Fig. 4. Sequential FHMM training algorithm pseudocode.

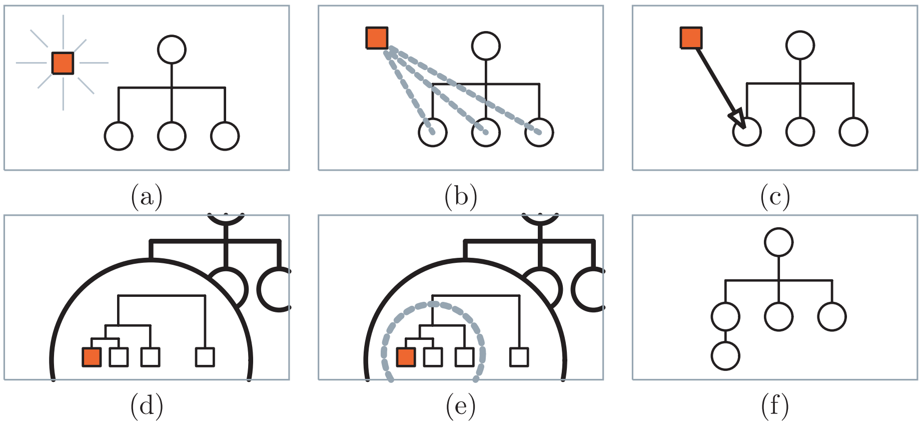

Fig. 5. Overview of the clustering algorithm (a square represents a data sequence, and a circle represents a group): (a) a new observation sequence is observed and encoded as an HMM; (b) the observation sequence is compared to existing groups via tree search; (c) the new sequence is placed in the closest existing group; (d) local clustering is performed on the modified group (zoomed in view of modified group); (e) a new subgroup is formed from similar motions in the modified group; (f) the subgroup is added to the tree as a child of the modified group.

cluster with sufficiently similar data is found, a child group is formed with this data subset (Figures 5(e) and (f)). The time series data of the motion examples forming the child group is then used to generate a single group model, which is subsequently used for both behavior recognition and generation. Therefore, the algorithm incrementally learns and organizes the motion primitive space, based on the robot's lifetime observations. The algorithm pseudocode is shown in Figure 7, while a schematic of the incremental memory structure formation is shown in Figure 6.

This algorithm allows the robot to incrementally learn and classify behaviors observed during continuous observation of a human demonstrator. The generation of a hierarchical structure of the learned behaviors allows for easier retrieval, and the automatic generation of the relationships between behaviors based on their similarity and inheritance. In addition, the robot's knowledge is organized based on the type of training received, so that the robot's knowledge will be most specialized in those areas of the behavior space where the most data has been observed.

3.1. Observation Sequence Encoding

Each newly acquired observation sequence is encoded into a HMM. In order to train the model, the HMM parameters, such as the number of states and the number of Gaussian mixtures must be selected. In previous work, we have used the AIC to select the best-fitting model (Kulic et al. 2007a). However, this approach can be time consuming, as it requires training each model candidate, and performing an exhaustive search of the model space. Using the AIC also does not consider the need for a better model when many similar motions need to be distinguished. Instead of using a fixed size model determined by

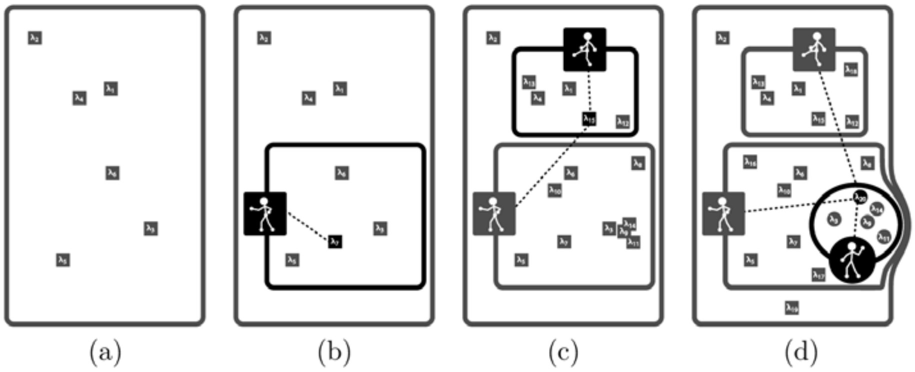

Fig. 6. Schematic illustration of the clustering algorithm: (a) initial state, when only one group is present; (b) a child group forms when enough similar examples are observed; (c) new observations are located into the closest group based on the distance between the new observation and the group model; (d) a higher order model is used in dense areas of the motion space.

```
1: procedure IncrementalCluster
2:     Step1 Encode observation sequence $O_i$ into an HMM $\lambda_i$
3:     Step2 Search the behavior tree for the closest group $\lambda_{Gj}$ to the current
       observation model $\lambda_i$, based on the inter-model distance
4:     Step3 Place $\lambda_i$ into the closest group $G_c$
5:     Step4 Perform clustering on all the exemplar motions within $G_c$
6:     Step5 If a sufficiently similar subgroup of motions is found, form a new
       group $G_n$, as a child of $G_c$, containing the observation sequences of the
       subgroup
7:     Step6 Using the observations sequences of the new subgroup, form the
       group model $\lambda_{Gn}$
8: end procedure
```

Fig. 7. Clustering algorithm pseudocode.

the AIC, in this paper we propose using a variable size model, where the number of dynamics chains in an FHMM model are increased based on the density of the motion exemplars in the relevant region of the motion space. With this approach, each motion is initially represented by a simple, single-chain, left-to-right HMM, with each state output model consisting of a single multivariate Gaussian distribution. Even though only a single observation sequence is used to generate the model, because the number of states is low relative to the number of time steps, a distribution about each state can be generated. The simple HMM model extracts a rough estimate of the key points of the trajectory. This model is then used for generating distances between motions for local clustering. If a better model is required, additional chain(s) are added as described below.

3.2. Intra model Distance Calculation

Once the newly observed behavior is encoded as an HMM, it is compared to existing groups (if any). Here, the distance between two HMMs can be calculated (Rabiner 1989) by

$$D(\lambda_1, \lambda_2) = \frac{1}{T}\left[\log P(O^{(2)}|\lambda_1) - \log P(O^{(2)}|\lambda_2)\right], \quad (6)$$

where λ_1, λ_2 are two HMM models, $O^{(2)}$ is an observation sequence *generated* by λ_2, and T is the length of the observation sequence. Since this measure is not symmetric, the average of the two intra HMM distances is used to form a symmetric measure:

$$D_s = \frac{D(\lambda_1, \lambda_2) + D(\lambda_2, \lambda_1)}{2}. \quad (7)$$

Note that Equation (7) is the pseudo-distance and not the distance in the strict sense since it does not satisfy the triangular equation. This distance measure is based on the relative log likelihood that a generated sequence is generated by one model, as compared to a second model. It represents a Kullback–Leibler distance between the two models. The formulation of the distance based on the model probability

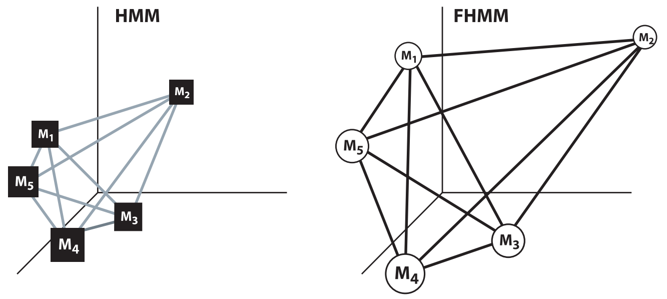

Fig. 8. Schematic comparing an HMM model vector space and an FHMM model vector space. The axes represent principal directions in the vector space (note the space does not necessarily need to be three-dimensional).

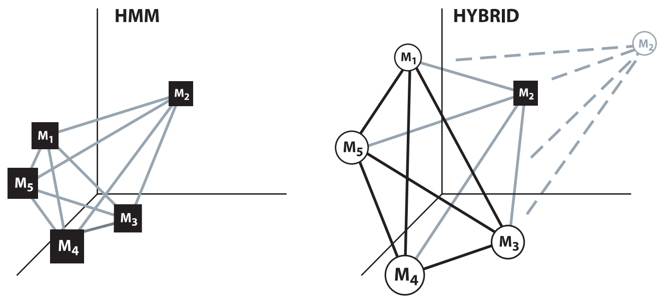

Fig. 9. Schematic comparing an HMM model vector space and a hybrid HMM–FHMM model vector space.

means that this measure can similarly be applied to FHMM models, by using the modified forward procedure (Ghahramani and Jordan 1997) to calculate the log likelihood, as well as used to compare FHMM and HMM models (Kulic et al. 2007a).

The distance measure quantifies the level of difficulty in discriminating between two models λ_1, λ_2. The distance measure can also be used to construct a motion pattern vector space by using multidimensional scaling (Takano et al. 2006). In this method, a vector space (named the *proto-symbol space*) is constructed such that the error between the actual distances and the distances in the vector space is minimized. By encoding more information about each pattern, FHMMs can improve the ability to discriminate between motion patterns, which can be especially useful when there are many similar motion patterns. Using the more detailed FHMM models increases the intra-model distances, as depicted conceptually in Figure 8. In addition, if the FHMM and HMM models of the same motion remain sufficiently similar, FHMM models may be combined with HMM models, by using FHMM models only in dense areas of the motion model space where better discriminative ability is required (shown in Figure 9).

The repository of known groups is organized in a tree structure, so that the new observation sequence does not need to be compared to all known behaviors. The comparison procedure is implemented as a tree search. At each node of the tree, the new observation sequence is compared to the leaves of that node. If the distance between the new observation sequence and one of the child nodes is sufficiently small, the search recurses to the most similar child node, otherwise the new observation sequence is added to the current node:

$$D_{\text{thresh}} = K_{\max GD} D^{G}_{\max}, \tag{8}$$

where D_{thresh} is the distance threshold at which a new observation sequence is considered for inclusion to a node, $K_{\max GD}$ is the multiplication factor applied, and $D^{G}_{\max}$ is the maximum intra observation distance for the given node. If the distance between the new observation and the cluster is larger than D_{thresh}, this cluster will not be considered as a possible match for the new observation sequence. If there are multiple candidate clusters, the new sequence is placed in the closest cluster. If there are no candidates, the new sequence is placed in the parent cluster. In the case of a new motion pattern which is completely dissimilar to any existing motion patterns, the motion pattern will be placed into the root node.

If the observation is being added to a group represented by a more detailed model (a model with a higher number of chains), the FHMM model is generated by adding additional chain(s) to the current representation.

Once the best matching cluster is selected for the newly observed sequence, the distance between the new observation and all the other observations in the cluster is calculated and added to the distance matrix stored for that node. This matrix is used for new cluster formation, as described in the next section.

3.3. Clustering and New Group Formation

When a new observation sequence is added to a cluster, a clustering procedure is invoked on that group, to determine if a subgroup may be formed. The complete link hierarchical clustering algorithm is used to generate the data clusters within a group (Jain et al. 1999). This is an agglomerative algorithm, where the data is initially placed in single element clusters, which are then successively joined based on the maximum intra element distance, so that the resulting output is a hierarchical tree describing the relationship between all the data points. Clusters can then be formed based on two criteria: the number of leaves in the cluster, and the maximum proximity measure of the potential cluster. Currently, both a minimum number of elements and a maximum distance measure are used. The maximum distance measure is based on the average of the inter motion distances in the cluster:

$$D_{\text{cutoff}} = \mu - K_{\text{cutoff}}\sigma, \tag{9}$$

where D_{cutoff} is the distance cutoff value (i.e. only clusters where the maximum distance is less than this value will be formed), and μ is the average distance between observations.

3.4. New Behavior Instantiation

If a new subgroup is generated in Step 5, a new group model is trained using the raw observation sequences from all the group elements. The structure of the new group model is determined based on the maximum intra observation distance for group, $D^G_{\max}$. Additional chains are added based on a simple threshold evaluation. In the experiments described below, the threshold value was determined manually; however, the threshold could also be determined automatically based on the inter-distance distribution within the group, or the decision could be based on testing for the presence of a multimodal distribution within the group. The generated model is subsequently used by the robot to generate behaviors. The group model replaces the individual observations in the parent node.

If one of the group elements allocated to the new cluster is already a group model, the generated motion sequence based on that model is used for the training. In this case, a modified form of the re-estimation formulas for multiple observation sequences (Rabiner 1989) is used. The algorithm is modified by over-weighting the group models, in order to account for the fact that there are multiple observation sequences stored in the generated model, and therefore more weight should be given to the group model, as compared to the individual observation sequences. In the experiments described below, over-weighting was used as it is the simplest and computationally least expensive approach. Alternatively, group exemplars could be generated by stochastically sampling multiple sequences from the group model, or by using the original data sequences comprising the group.

3.4.1. Superseding

A special case that may occur during new cluster formation is that group models in a parent group may become targeted for inclusion in a newly formed subgroup. Once a new subgroup is formed, the behaviors forming the subgroup are removed, and replaced with a single, group model. Thereafter, this group model is treated exactly the same as a single behavior model, except that it is over-weighted during cluster group HMM training. Therefore, as more behaviors get added to the group, occasionally the group model may be included in a new subgroup. In this case, the tree structure is also updated to insert the newly formed subgroup as the parent of the group model which has been subsumed by the new group.

3.5. Computational Efficiency and Memory Usage

The developed algorithm is suitable for on-line behavior acquisition, as the computational requirements are significantly lower as compared to a global clustering approach. There is no need for an exhaustive search in the model space, as the model size is adjusted on-line based on the similarity between the new motion and previously known motions. A larger model is only utilized when motions are similar to each other, and better discriminative ability is required. Each new sequence is compared to known sequences via tree search, reducing the number of comparisons required. Once the closest node is found, the computation time for node clustering is constant (only the models in the closest node are clustered). Each cluster is limited to a maximum number of observation sequences, $N_{\max}$. If a new observation sequence is being added to a cluster which already contains $N_{\max}$, an old observation sequence is selected for removal from the cluster, before adding the new observation. Currently, the observation sequence to be removed is selected based on FIFO (first in, first out).

In addition, to conserve memory resources, it is also possible to amalgamate leaf clusters into a single observation (i.e. do not store any of the constituent observation sequences for a leaf cluster, but only the resulting group HMM). However, following amalgamation, it would no longer be possible to subdivide that cluster, i.e. it would be considered as a single observation in the parent cluster. Therefore, amalgamation could be performed once the tree hierarchy had reached a certain depth level, or once the distance between cluster elements has become sufficiently small.

3.6. Deterministic Motion Generation

Once a cluster node has been formed, the group HMM for the node constitutes the abstraction of the motion primitive. This model is then used to generate a motion trajectory for the robot to execute.

When the generated motion sequence is to be used for robot motion commands, we do not want to introduce the noise characteristics abstracted by the HMM model. In this case, we use a greedy policy to estimate the optimum state sequence. First, for each chain m, the starting state q^m_0 is selected by choosing the highest value from the initial state probability distribution. At each state, the state duration is calculated based the state transition matrix,

$$\bar{d}^m_i = \frac{1}{1 - a^m_{ii}}. \tag{10}$$

Following $\bar{d}^m_i$ samples in state i, the next state is selected by a greedy policy from the state transition matrix, excluding the a^m_{ii} probability. If the model type is front-to-back, the algorithm iterates until the final state is reached, otherwise the state sequence is generated for the specified number of time steps. Once the state sequence has been generated for each chain,

```
1:  procedure GENERATE
2:      for $m \leftarrow 1, M$ do
3:          Initialize
4:          $q_0^m \leftarrow argmax_i(\pi_m(i))$
5:          while $t < T, i \neq i_{terminal}$ do
6:              $\bar{d}_i^m \leftarrow \frac{1}{1-a_{ii}^m}$
7:              $q_{t:t+\bar{d}_i^m} \leftarrow q_t$
8:              $q_{t+1} \leftarrow argmax_i a_{ij}^m, i \neq j$
9:          end while
10:     end for
11:     for $t \leftarrow 0, T$ do
12:         $y_t \leftarrow \sum_{m=1}^{M} B^m(q_t^m)$
13:     end for
14: end procedure
```

Fig. 10. Greedy generation algorithm pseudocode.

the output sequence is calculated by summing the contribution from each chain at each time step, based on that chain's current state value. The generation algorithm pseudocode is shown in Figure 10. This algorithm selects the state sequence by a local greedy policy over the state transition algorithm. Alternatively, if a motion most similar to a recently observed motion is required, the optimal state sequence could be generated by using the Viterbi algorithm, as described by Lee and Nakamura (2005).

After the trajectory is generated, some low-pass filtering or smoothing is required as a post-processing step to eliminate the artifacts caused by discrete state switching and generate a smooth trajectory for use as a command input.

4. Experiments

The experiments described below have been performed on a data set containing a series of nine different human movement observation sequences obtained through a motion capture system (Kadone and Nakamura 2005). The data set contains joint angle data for a 20 degrees of freedom (DoF) humanoid model from multiple observations of walking (WA, 28 observations), cheering (CH, 15 observations), dancing (DA, 7 observations), kicking (KI, 19 observations), punching (PU, 14 observations), sumo leg raise motion (SL, 13 observations), squatting (SQ, 13 observations), throwing (TH, 13 observations), and bowing (BO, 15 observations). Figure 11 (see also Extension 1) shows an example of a walking motion from the data set. Sample animations of each of the other motion types can be seen in Extension 1. Since the data set includes only the joint angle data of the rotational joints and does not include the position and rotation of the 6 DoF body joint, the torso of the humanoid model is held fixed in the animations.

In the first set of experiments, the performance of the FHMMs is compared with conventional HMMs for recognition and generation, and the validity of using FHMMs and HMMs selectively is confirmed. In the second set of experiments, the incremental clustering and organization algorithm described in Section 3 is tested.

4.1. HMM to FHMM Comparison

To compare the performance of conventional HMMs and FHMMs, a set of nine HMM and FHMM models each are trained on the data, one for each motion type. In addition, the performance of sequentially trained FHMMs is compared to FHMMs trained with the exact algorithm. For each type of motion, four training procedures were applied to the FHMM: the exact training method as developed by Ghahramani and Jordan (1997), sequential training using Gamma-based residual error (Equation (2)), sequential training using Viterbi-based residual error (Equation (3)), and sequential training using generation-based residual error (Equation (4)). In this set of tests, the data is manually segmented into the nine categories described above. Each FHMM consists of a two chains of left-to-right (Bakis type) hidden Markov chains of 15 states each. Example animations of each motion type generate from the sequentially trained FHMM model can be seen in Extension 1. Since the data set includes only the joint angle data of the rotational joints and does not include the position and rotation of the 6 DoF body joint, the torso of the humanoid model is held fixed in the animations.

The corresponding HMMs are also front-to-back type, and contain an equivalent (15×15) number of states. For both model types, the covariance matrix was constrained to be diagonal during training, and the minimum covariance was constrained to 0.001, to avoid numerical underflow/overlow during training. Each model was trained on five randomly selected exemplars of a motion type. The learning algorithm for each model was run for 100 iterations. Each model was then tested by computing the log likelihood of observing the following observation sequences: an example from the training set, a randomly drawn example of the same motion outside the training set, and an example of a different motion. This testing procedure was repeated for 100 trials. At the start of each trial, the state transition parameters were initialized to random values. The mean and variance parameters in the output distribution model were initialized by calculating the mean values and covariance of the output vector over all the training data. The means were initialized by sampling from a Gaussian distribution with the data based means and covariance. The average log likelihood and standard deviation for each test case are shown in Tables 1 and 2, for the FHMM models and HMM models, respectively. Figure 12 shows a graphical comparison of the recognition results between the HMM and FHMM models for the training and novel data.

As can be seen from the results in Tables 1 and 2, HMMs perform well at recognizing data from the training set, but

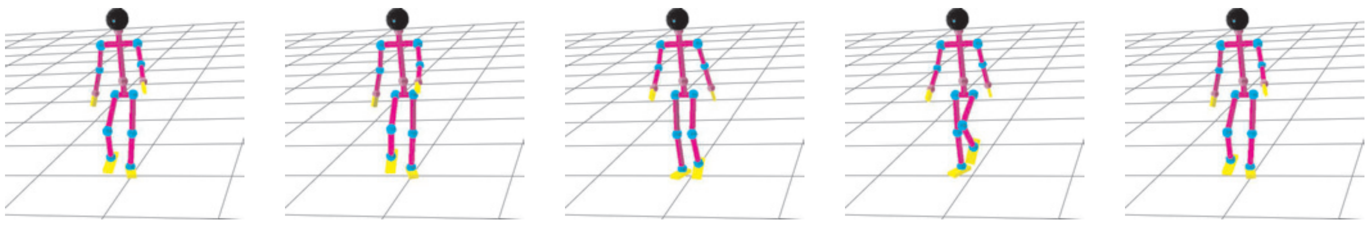

Fig. 11. Sample walking motion (see also Extension 1).

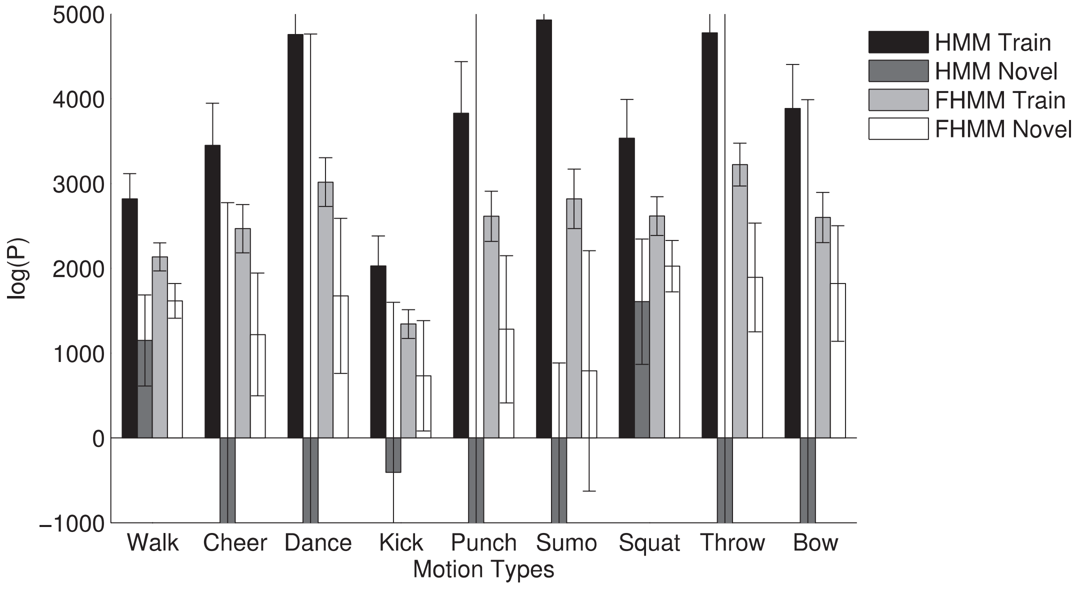

Fig. 12. Comparison between the HMM and FHMM recognition results (in terms of log likelihood) for training and novel data.

Table 1. FHMM recognition results, exactly trained model. The training column indicates the average log likelihood and standard deviation over the 100 test cases for a randomly selected exemplar out of the training set, the novel column indicates the log likelihood distribution for an example of the same motion, but outside of the training set, and the different column indicates the log likelihood distribution for an example of a different motion.

Motion type	Training	Novel	Different
Walk	2,134 ± 165	1,616 ± 205	−66,589 ± 15,158
Cheer	2,468 ± 284	1,220 ± 724	−60,908 ± 19,899
Dance	3,016 ± 288	1,675 ± 913	−60,922 ± 15,016
Kick	1,344 ± 170	733 ± 651	−48,060 ± 17,208
Punch	2,613 ± 295	1,281 ± 868	−52,786 ± 23,188
Sumo leg	2,818 ± 350	791 ± 1,418	−49,262 ± 20,734
Squat	2,617 ± 229	2,025 ± 302	−58,060 ± 17,283
Throw	3,223 ± 254	1,893 ± 641	−48,517 ± 21,690
Bow	2,599 ± 295	1,821 ± 681	−54,118 ± 18,877

Table 2. HMM recognition results. The training column indicates the average log likelihood and standard deviation over the 100 test cases for a randomly selected exemplar out of the training set, the novel column indicates the log likelihood distribution for an example of the same motion, but outside of the training set, and the different column indicates the log likelihood distribution for an example of a different motion.

Motion type	Training	Novel	Different
Walk	2,818 ± 297	1,150 ± 536	−48,973 ± 25,913
Cheer	3,448 ± 497	−2,461 ± 5,236	−50,147 ± 25,914
Dance	4,756 ± 687	−3,234 ± 7,996	−59,168 ± 15,563
Kick	2,028 ± 353	−405 ± 2,003	−19,262 ± 17,553
Punch	3,828 ± 611	−8,497 ± 18,242	−53,006 ± 19,779
Sumo leg	4,929 ± 1068	−5,727 ± 6,611	−45,079 ± 21,941
Squat	3,535 ± 457	1,606 ± 738	−31,098 ± 26,645
Throw	4,776 ± 1,454	−15,468 ± 26,723	−53,540 ± 20,551
Bow	3,884 ± 519	−1,980 ± 5,968	−52,923 ± 24,598

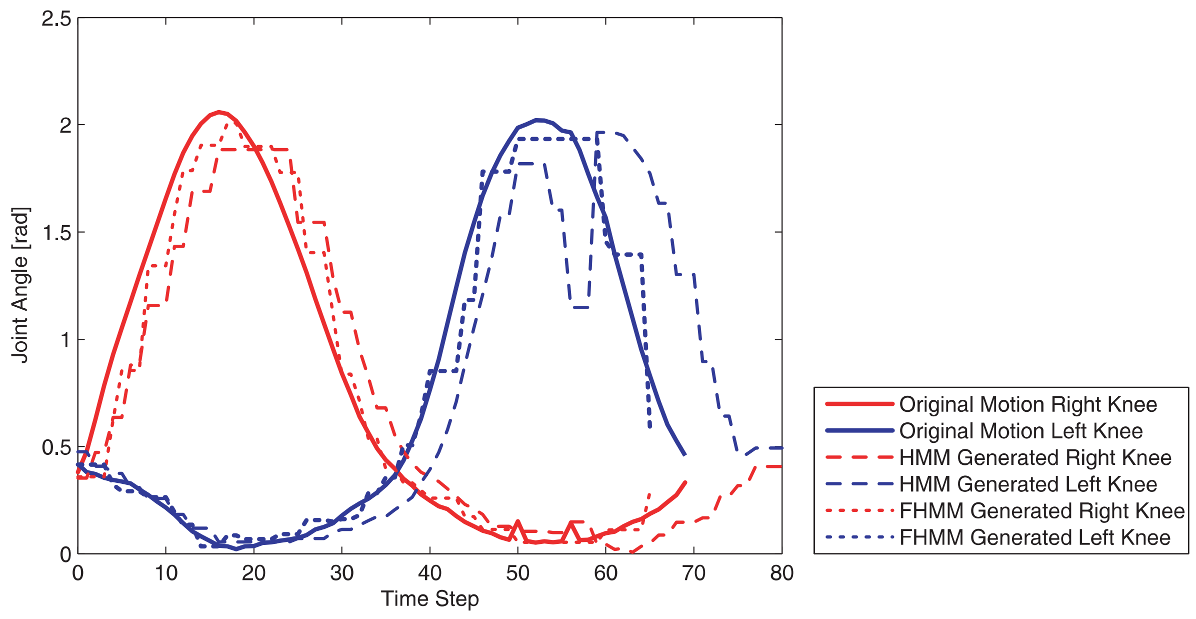

Fig. 13. Comparison of generation results of the HMM (15 × 15 states) and FHMM (two chains of 15 states) for the knee joints during a walking motion, prior to applying any post processing.

show a significant drop in performance when recognizing new data, indicating over-fitting. On the other hand, FHMMs demonstrate significantly better generalization, and show a much smaller drop in performance when recognizing new motions of the same type as compared to the HMM. Both models are equally good at rejecting data from a different motion sequence.

The FHMM is better at reliably generating motion sequences which have good fidelity to the trained data, while the HMM generated motions show a very large variability, due to the problem of over-fitting. Figure 13 shows an example of generation results for the left and right knee joints during a walking motion, before any post-processing of the trajectory has been applied. One of the walking motions used during training of the models is shown for comparison. Figure 14 shows an example of a walking motion generated by the walk FHMM. As can be seen in Figure 13, in the motion of the left knee, the HMM fitted state outputs at two different peaks,

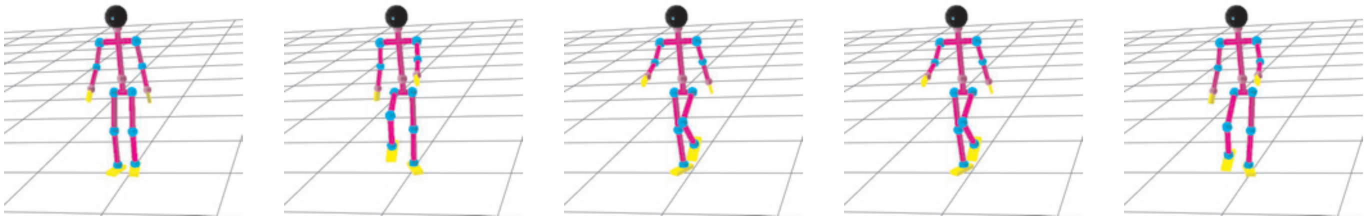

Fig. 14. FHMM generated walking motion (see also Extension 1).

Table 3. FHMM Recognition results, model trained sequentially using the Gamma method. The training column indicates the average log likelihood and standard deviation over the 100 test cases for a randomly selected exemplar out of the training set, the novel column indicates the log likelihood distribution for an example of the same motion, but outside of the training set, and the different column indicates the log likelihood distribution for an example of a different motion.

Motion type	Training	Novel	Different
Walk	2,271 ± 90	1,930 ± 154	−58,235 ± 19,017
Cheer	2,632 ± 151	1,731 ± 526	−44,197 ± 23,454
Dance	3,290 ± 327	2,414 ± 709	−52,712 ± 19,468
Kick	1,564 ± 146	1,081 ± 427	−27,053 ± 14,655
Punch	2,735 ± 262	1,736 ± 732	−41,404 ± 17,690
Sumo leg	2,936 ± 317	1,505 ± 1,617	−23,740 ± 15,093
Squat	2,863 ± 158	2,453 ± 171	−38,365 ± 13,790
Throw	3,331 ± 177	2,384 ± 469	−27,690 ± 17,980
Bow	2,851 ± 268	2,260 ± 513	−26,450 ± 15,528

likely due to the same peak executed at different speeds in the training set, indicating over-fitting.

Tables 3, 4 and 5 show the results for a FHMM model trained sequentially, using the Gamma values, Viterbi state sequence, and the generated output from preceding chains to calculate the residual error, respectively. Figure 15 graphically shows the comparison between the exact and sequentially trained models for training and novel data, while Figure 16 illustrates the differences in performance of the sequential training algorithm, comparing the performance for the different methods for generating the residual error. All of the test cases were carried out on a 2.66 GHz Intel Xeon CPU. The average time per training cycle for the exactly trained FHMMs was 2.4 s, compared with 0.1 s for the sequentially trained models. This is due to the fact that the proposed sequential training approach executes only a single pass of the generalized-backfitting algorithm (Jacobs et al. 2002), consisting of a single expectation-minimization for each chain of the FHMM. Among the three sequential training approaches, the sequential training using Gamma-based residual error (Equation (2)) achieved the best results, however the difference was not significant. As can be seen from Tables 1 and 3, the sequentially trained models achieve comparable and sometimes improved results over the Ghahramani and Jordan (1997) approach, while requiring significantly less training time. The improvement in performance is likely due to the fact that Ghahramani and Jordan assume a single, constant covariance matrix across all states, whereas in the proposed approach separate covariance matrices at each state are estimated, providing a more accurate estimate of the variance.

It should be noted that the sequentially trained FHMM is not equivalent to utilizing two independent HMMs during recognition. (i.e. with one independent HMM trained on the training data set, and the second independent HMM trained on the error between the data set and the output of the first HMM). While the two FHMM chains are independent during *generation*, and thus equivalent to the two independent HMMs, during recognition the FHMM formulation induces a dependence between the two chains. This is because the likelihood is being evaluated *given the observed data sequence*, thus introducing a dependence between the two chains at each observation time step. Table 6 shows the recognition results for the two independent HMMs formulation. The final likelihood of each motion sequence tested is calculated as a product of the likelihoods from each of the HMM models (independent HMM models assumption). Figure 17 shows the comparison

Table 4. FHMM recognition results, model trained sequentially using the Viterbi method. The training column indicates the average log likelihood and standard deviation over the 100 test cases for a randomly selected exemplar out of the training set, the novel column indicates the log likelihood distribution for an example of the same motion, but outside of the training set, and the different column indicates the log likelihood distribution for an example of a different motion.

Motion type	Training	Novel	Different
Walk	$2{,}240 \pm 94$	$1{,}922 \pm 156$	$-55{,}476 \pm 22{,}699$
Cheer	$2{,}502 \pm 227$	$1{,}895 \pm 390$	$-41{,}743 \pm 18{,}574$
Dance	$3{,}319 \pm 280$	$2{,}427 \pm 642$	$-52{,}484 \pm 16{,}430$
Kick	$1{,}468 \pm 143$	$1{,}024 \pm 443$	$-19{,}499 \pm 13{,}832$
Punch	$2{,}666 \pm 249$	$1{,}753 \pm 814$	$-39{,}252 \pm 19{,}056$
Sumo leg	$2{,}806 \pm 259$	$1{,}260 \pm 1{,}609$	$-18{,}246 \pm 11{,}622$
Squat	$2{,}812 \pm 154$	$2{,}423 \pm 176$	$-37{,}256 \pm 13{,}430$
Throw	$3{,}310 \pm 197$	$2{,}371 \pm 479$	$-28{,}694 \pm 17{,}366$
Bow	$2{,}751 \pm 291$	$2{,}155 \pm 471$	$-22{,}729 \pm 14{,}540$

Table 5. FHMM recognition results, model trained sequentially using the Generated method. The training column indicates the average log likelihood and standard deviation over the 100 test cases for a randomly selected exemplar out of the training set, the novel column indicates the log likelihood distribution for an example of the same motion, but outside of the training set, and the different column indicates the log likelihood distribution for an example of a different motion.

Motion type	Training	Novel	Different
Walk	$2{,}199 \pm 107$	$1{,}886 \pm 180$	$-51{,}429 \pm 21{,}074$
Cheer	$2{,}562 \pm 204$	$1{,}809 \pm 491$	$-38{,}479 \pm 21{,}639$
Dance	$3{,}219 \pm 272$	$2{,}285 \pm 692$	$-53{,}262 \pm 17{,}050$
Kick	$1{,}407 \pm 177$	937 ± 419	$-16{,}557 \pm 12{,}294$
Punch	$2{,}573 \pm 260$	$1{,}709 \pm 752$	$-25{,}160 \pm 16{,}495$
Sumo leg	$2{,}669 \pm 309$	$1{,}459 \pm 1{,}267$	$-15{,}164 \pm 9{,}466$
Squat	$2{,}777 \pm 151$	$2{,}409 \pm 228$	$-30{,}647 \pm 10{,}673$
Throw	$3{,}243 \pm 160$	$2{,}330 \pm 472$	$-30{,}857 \pm 18{,}434$
Bow	$2{,}603 \pm 340$	$2{,}214 \pm 421$	$-21{,}414 \pm 13{,}915$

between the sequentially trained FHMM results and the two independent HMMs. For both the independent HMMs and for the sequentially trained FHMMs, the γ method (Equation (2)) is used to calculate the residual error.

As can be seen from Figure 17, using independent HMMs for recognition can give better results for some motion types, but increases the variability in performance for all motion types, and fails to correctly characterize some motions. Motions which are most similar and have little inter-group variability (for example, the walk motion) are better characterized by the independent HMMs model, while longer motions and motions with more temporal and spatial variability (for example, the sumo leg raise, or the bow motions) cannot be characterized appropriately by using independent HMMs.

In the second set of experiments, the increase in accuracy and discrimination power of using an FHMM was compared against using a single-chain HMM with a small number of states. The same set of FHMMs was used as in the first set of experiments above, while the HMMs consisted of left-to-right models with 15 states each. Figure 18 shows a comparison between the FHMM and the low state number HMM before any trajectory post-processing has been applied. As can be seen in Figure 18, due to the higher number of states available to represent the motion, FHMMs achieve better spacial accuracy compared to a single-chain HMM model.

Table 7 shows the intra-model distances between each of the HMM models, and Table 8 shows the equivalent intra-model distances between the FHMM models. The distances

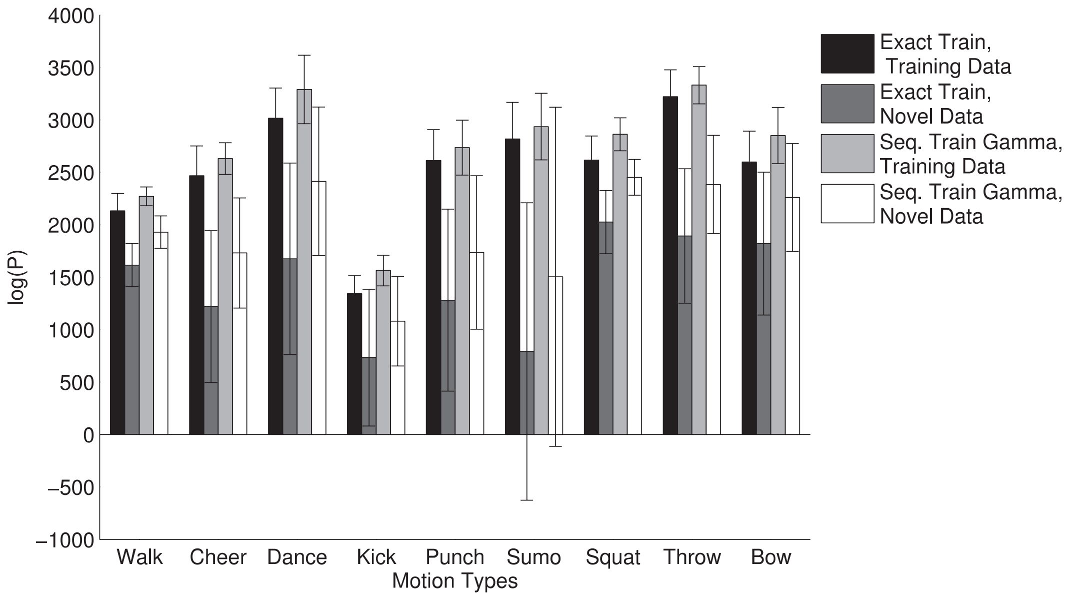

Fig. 15. Comparison between the exactly trained and sequentially trained (Gamma method) FHMM recognition results (in terms of log likelihood) for training and novel data.

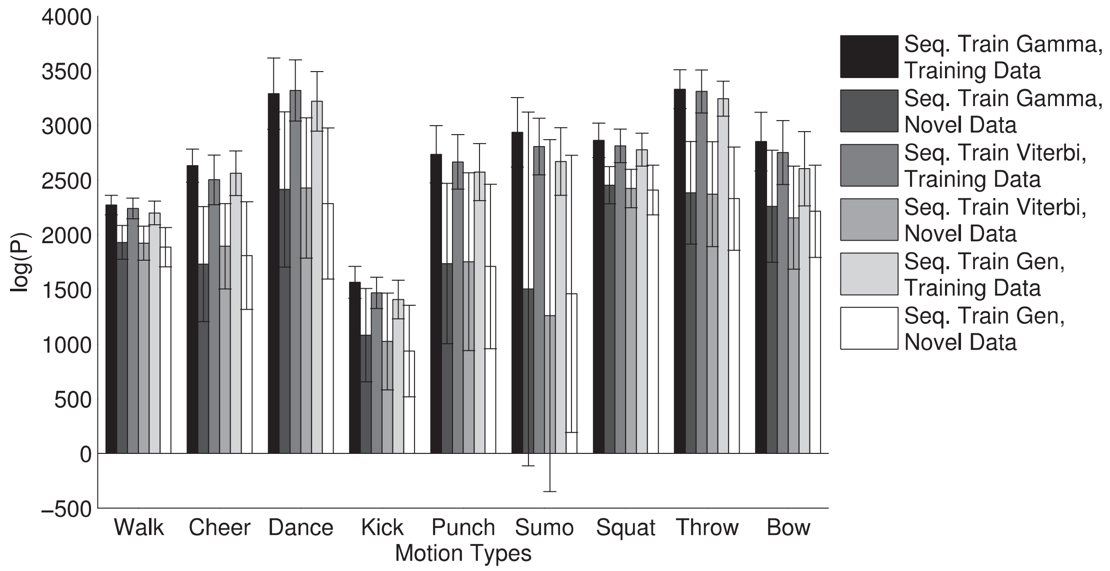

Fig. 16. Comparison between the different methods of generating the residual error for sequentially trained FHMM models. Recognition results (in terms of log likelihood) for training and novel data.

are calculated according to Equation (7). The average intra-model distance between HMM models is 377, while the average intra-model distance between FHMM models is 467. The use of FHMMs increases the models discriminative ability by enlarging the distances between each model in the proto-symbol space (Takano et al. 2006).

FHMMs can also be compared with HMMs using the probabilistic distance measure (Equation (7)). Table 9 compares the distances between the FHMM and HMM models. As expected, the inter-model distance for the same motion type is very small (average 10.2), validating the use of a mixture of HMM and FHMM models.

Table 6. Recognition results for two independent HMM chains, one trained on the full training data, and the second trained on the residual error (calculated via the Gamma method) between the data and the first chain. The training column indicates the average log likelihood and standard deviation over the 100 test cases for a randomly selected exemplar out of the training set, the novel column indicates the log likelihood distribution for an example of the same motion, but outside of the training set, and the different column indicates the log likelihood distribution for an example of a different motion.

Motion type	Training	Novel	Different
Walk	$4{,}373 \pm 183$	$3{,}957 \pm 281$	$-62{,}765 \pm 18{,}652$
Cheer	$4{,}569 \pm 632$	$3{,}498 \pm 976$	$-61{,}426 \pm 23{,}659$
Dance	$6{,}272 \pm 713$	$4{,}865 \pm 1{,}361$	$-56{,}671 \pm 17{,}230$
Kick	$2{,}546 \pm 801$	$1{,}510 \pm 1{,}636$	$-39{,}510 \pm 17{,}784$
Punch	$4{,}918 \pm 687$	$3{,}564 \pm 1{,}260$	$-52{,}062 \pm 23{,}262$
Sumo leg	$4{,}714 \pm 1{,}285$	$1{,}730 \pm 8{,}280$	$-37{,}576 \pm 20{,}168$
Squat	$5{,}281 \pm 453$	$4{,}553 \pm 716$	$-64{,}454 \pm 29{,}019$
Throw	$6{,}008 \pm 573$	$4{,}883 \pm 916$	$-60{,}228 \pm 26{,}387$
Bow	$1{,}490 \pm 11{,}256$	$1{,}334 \pm 8{,}834$	$-66{,}820 \pm 20{,}039$

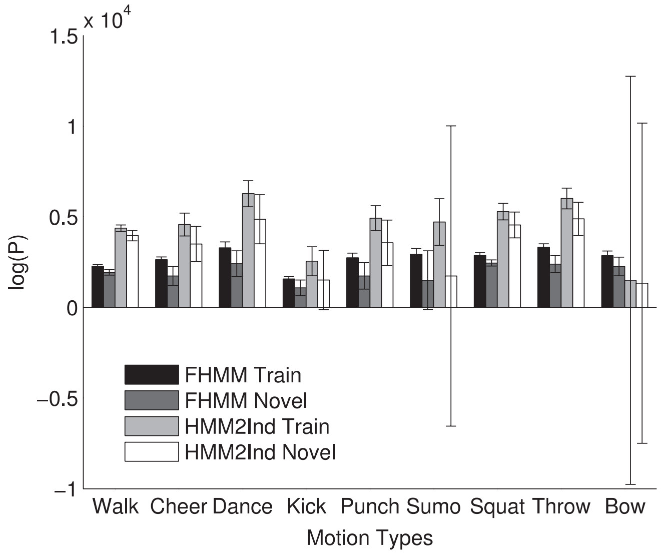

Fig. 17. Comparison between the sequentially trained (Gamma method) FHMM and two independent HMM recognition results (in terms of log likelihood) for training and novel data.

4.2. Automated Clustering and Hierarchy Formation

4.2.1. Random Order Motion Presentation

In the next set of tests, the incremental clustering and hierarchy formation algorithm is validated. In the first set of the experiments, motion sequences are presented to the algorithm in random order. Motion sequences are presented one at a time, simulating on-line, sequential acquisition. After each motion is presented, the algorithm is executed, performing incremental clustering. In each simulation performed, the algorithm correctly segments the behaviors such that the resulting leaf nodes represent the grouping that would be obtained with an off-line method. Out of 100 simulation runs performed at each of the parameter settings, there was no cases of misclassification at the leaf nodes, showing that the final segmentation is robust to presentation order. Here, misclassification is defined as a false positive error (for example, a walk motion being misclassified as a punch motion). However, there were cases of false negative errors, where a motion which should have been recognized as a known motion was instead placed into a non-leaf node (for example, the root node). Experiments using only a single-chain HMMs, as well as experiments using adaptable models, were performed. For the case when single-chain HMMs were used, the appropriate model size was selected via the AIC. Sample segmentation results for the single-chain HMMs are shown in Figures 19, 20, and 21. Note that the actual order of node formation will vary depending on the motion presentation order. The average rate of false negative errors and the standard distribution of the false negative errors is shown for each motion in Tables 10 and 11. As noted before, no false positive errors were observed in any of the experiments performed.

The algorithm parameter K_{cutoff} (the parameter which controls when a new cluster is formed from an existing cluster) determines the resultant tree structure. A high value for K_{cutoff} (i.e. only clusters composed of a tight data set are formed) tends to result in a flat tree structure (as shown in Figure 19, while low values of K_{cutoff} result in a deep tree structure, as shown in Figure 20. As the cluster formation parameter is re-

Table 7. HMM intra-model distances.

	WA	CH	DA	KI	PU	SL	SQ	TH	BO
WA	0								
CH	488	0							
DA	728	723	0						
KI	485	444	526	0					
PU	713	696	423	382	0				
SL	435	513	469	400	463	0			
SQ	554	619	534	594	576	295	0		
TH	702	648	454	421	78	447	662	0	
BO	562	477	448	469	552	336	538	561	0

Table 8. FHMM intra-model distances.

	WA	CH	DA	KI	PU	SL	SQ	TH	BO
WA	0								
CH	659	0							
DA	740	734	0						
KI	638	658	604	0					
PU	741	740	718	636	0				
SL	528	699	689	720	674	0			
SQ	740	739	729	688	642	737	0		
TH	732	729	567	580	115	698	696	0	
BO	740	739	504	678	643	737	695	638	0

Table 9. Inter model distances.

	FHMM models								
HMM models	WA	CH	DA	KI	PU	SL	SQ	TH	BO
WA	8.3								
CH	647	11.2							
DA	734	731	8.3						
KI	612	636	553	8.8					
PU	736	734	652	442	11.8				
SL	509	680	675	657	645	16.5			
SQ	735	725	705	617	596	497	6.8		
TH	726	722	512	419	81	493	682	9.7	
BO	735	645	475	576	587	517	576	607	10.6

laxed, deeper trees tend to be formed. However, the resulting tree structure tends to be dependent on the presentation order. In the case of a high cutoff value (see Figure 19), the resulting tree structure is flat, and fairly insensitive to presentation order. The resulting structure is consistent with off-line clustering result. In about 9% of cases, the "dance" group fails

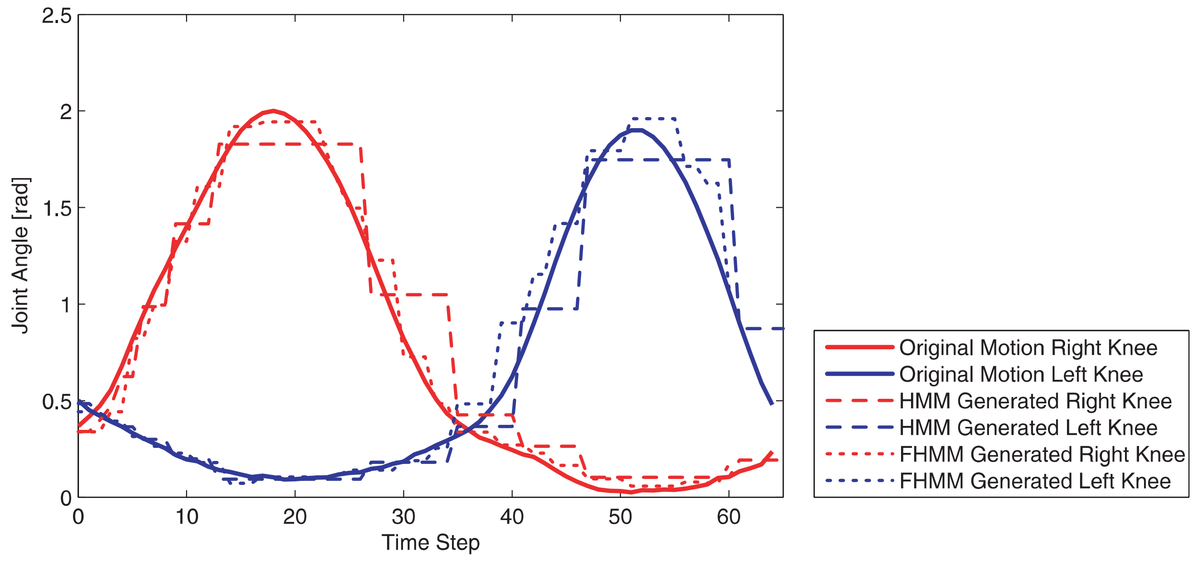

Fig. 18. Comparison of generation results of the HMM (15 states) and FHMM (two chains of 15 states) for the knee joints during a walking motion, prior to applying any post processing.

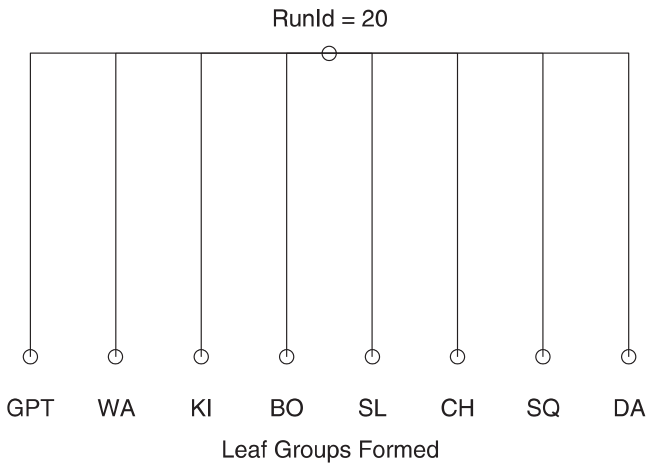

Fig. 19. Sample segmentation result, $K_{\text{cutoff}} = 1.2$.

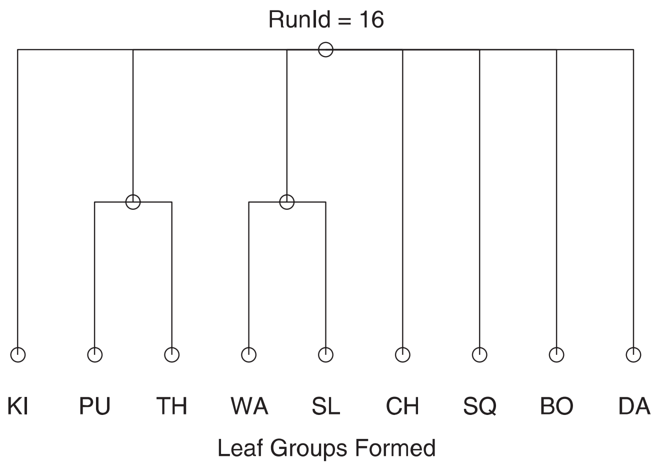

Fig. 20. Sample segmentation result, $K_{\text{cutoff}} = 0.9$.

to form, in contrast to the off-line clustering result, since this group contains the least examples. At the high cutoff value, the punch and throw motions are too similar to subcluster, resulting in a single hybrid generated motion (indicated as PU/TH in Figure 19). This is also indicated in the high false negative error rates for the punch and throw motions in Table 10, as the motions do not tend to be recognized as belonging to a distinct motion type, but are instead placed in the group node. The generated motion resulting from that subcluster is shown in Figure 22 (see also Extension 1). As can be seen in the figure, the motion is an averaging of the two motions.

When low values of K_{cutoff} are used, nodes are quicker to form, and the resulting tree structure becomes more dependent on presentation order. The similarity level at which nodes will form is highly dependent on presentation order. Figures 20 and 21 show two examples of different tree structures formed, from two simulation runs. Note that the identified leaf nodes remain the same. In addition, using the lower cutoff value makes it easier to subdivide the similar throw and punch motions. This can be seen from the lower false negative rate in Table 11, as significantly more of the punch and throw motions are correctly recognized. Note that some punch throw motions still remain difficult to recognize, and are instead placed in the punch/throw parent node, which is classified as a false negative error in Table 11. Even though the cutoff level was the same for both experiments, the similarity level of the nodes

Table 10. False negative errors, $K_{\text{cutoff}} = 1.2$ (an error rate of 1 indicates that all motions fail to be recognized).

	WA	CH	DA	KI	PU	SL	SQ	TH	BO
False negatives (%)	0.0068	0.0273	0.0843	0.0621	0.9879	0.0254	0.0377	0.9900	0.0047
Standard deviation	0.0316	0.1447	0.2725	0.0789	0.0891	0.1247	0.1332	0.1000	0.0358

Table 11. False negative errors, $K_{\text{cutoff}} = 0.9$ (an error rate of 1 indicates that all motions fail to be recognized).

	WA	CH	DA	KI	PU	SL	SQ	TH	BO
False negatives (%)	0.0168	0.0407	0.1929	0.0526	0.4093	0.0492	0.0546	0.5638	0.0360
Standard deviation	0.0786	0.0918	0.3934	0.0876	0.2240	0.1744	0.1084	0.3585	0.0896

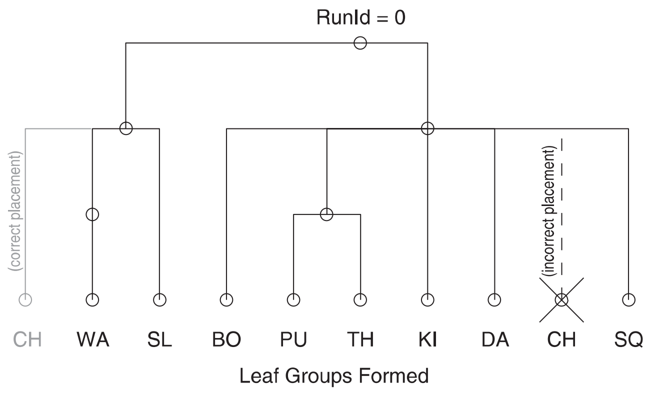

Fig. 21. Sample segmentation result, $K_{\text{cutoff}} = 0.9$.

formed differed, based on the presentation order. The result in Figure 20 is consistent with global clustering, while, in the result shown in Figure 21, one node is incorrectly assigned. The CH node is incorrectly assigned to the PU/TH/KI/SQ branch of the tree, whereas global clustering would have assigned the CH node to the WA/SL branch. This type of error is due to the local nature of the algorithm, i.e. clustering is being performed when only a part of the data set is available. Therefore, there is a tradeoff when selecting the K_{cutoff} value between facilitating quick node formation and differentiation and introducing misclassifications in the hierarchy tree. However, since the leaf nodes are identified correctly regardless of the K_{cutoff} value, a slower rate tree correction algorithm may be periodically applied, to reposition leaf nodes to the correct branch as more data becomes available.

When single-chain HMM models are used, at high levels of K_{cutoff}, the similar motions of punch and throw cannot be distinguished. However, if both HMM and FHMM models are used, such that FHMM models are inserted into a dense region of the motion space, as described in Section 3.2, better discrimination ability can be achieved, given the same number of training examples. A sample result of the clustering performance using the adaptable models is shown in Figure 23. The adaptable model can distinguish between similar motions TH and PU, whereas those motions cannot be distinguished when only single-chain HMM models are used (see Figure 19). The false negative error rate and standard deviation is shown in Table 12. As can be seen from this table, the adaptable models achieve an improved error rate, particularly for the punch and throw motions. As in the previous experiments, no false positives were recorded over the 100 experiments executed with the adaptable models.

4.2.2. Non-random Order Motion Presentation

The incremental clustering algorithm was also tested to simulate the scenario when motions are not being presented in random order. This type of non-random presentation could occur when multiple examples of the same motion are presented contiguously. In this test case, all motions of each type were presented sequentially as a group, starting with all the walk examples, and following with cheer, dance, kick, punch, sumo leg raise, squat, throw, and bow examples. The algorithm is also able to successfully cluster the motions in this case. As in the previous test cases, no misclassifications (false positives) occur at the leaf nodes. Figure 24 shows the clustering results, and Table 13 shows the rate of false negatives in this case. The performance is comparable to the random order case.

However, in the case of the non-random presentation order, the formation of the first child node is significantly delayed. The first node formed (the walk node) does not form until after

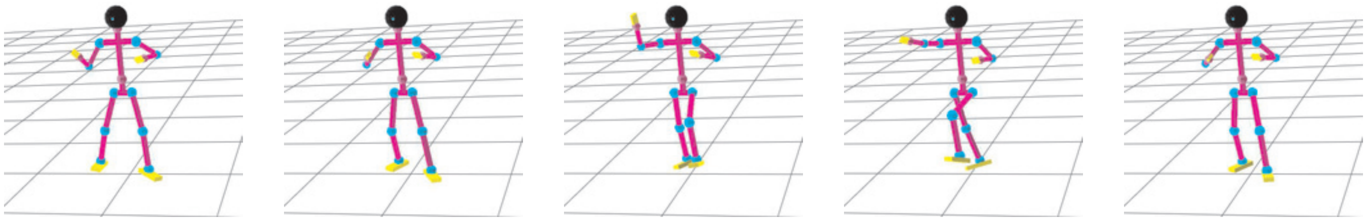

Fig. 22. Generated hybrid punch/throw motion (see also Extension 1).

Table 12. False negative errors, adaptable models (an error rate of 1 indicates that all motions fail to be recognized).

	WA	CH	DA	KI	PU	SL	SQ	TH	BO
False negatives (%)	0.0025	0.0133	0.0714	0.0632	0.2914	0.0100	0.0615	0.3831	0.0253
Standard deviation	0.0105	0.0519	0.2564	0.0733	0.1789	0.1000	0.1289	0.3065	0.0575

Table 13. False negative errors, non-random presentation order (an error rate of 1 indicates that all motions fail to be recognized).

	WA	CH	DA	KI	PU	SL	SQ	TH	BO
False negatives (%)	0.11	0	0	0	0.42	0	0	0.38	0

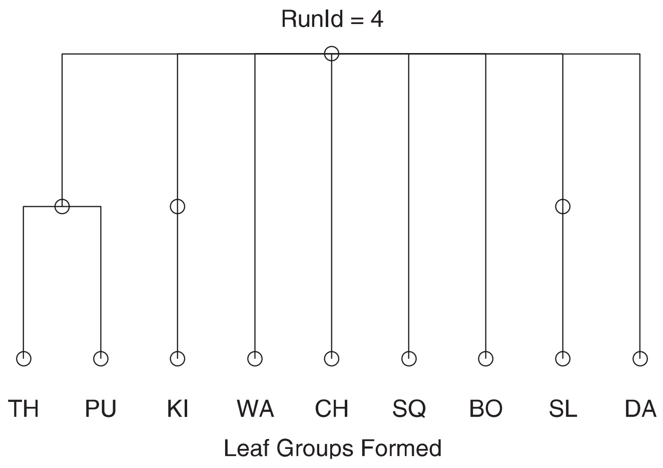

Fig. 23. Sample segmentation results when using adaptable models.

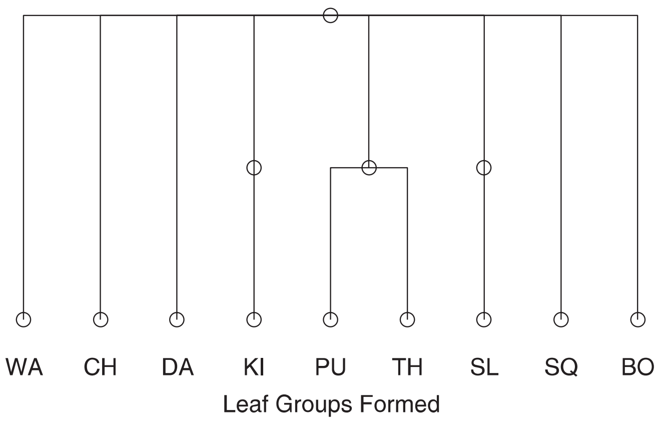

Fig. 24. Sample segmentation results when motions are presented in non-random, contiguous order. All examples of the walk motions were presented first, followed by all examples of cheer, dance, kick, punch, sumo leg raise, squat, throw, and bow.

the first non-walk example is presented. In addition, if the first presented motion has significant variability (this was not the case for the walk motion, but is the case for the kick motion, for example), the first node formed may not include all the presented motion exemplars, but only a subset of the most similar motions. Following subsequent presentations of examples of a different motion, a second node forms containing all the examples of the first presented motion and superseding the smaller node, as described in Section 3.4.1. From this point forward (i.e. once there are at least four known examples of two different motions), the algorithm performance is comparable to the random results described above. Therefore, this type of motion presentation initially delays learning, but once more than one type of motion is observed, the algorithm is unaffected by non-random presentation order.

This delay in the formation of the first node can be addressed by pre-initialization of the root node. In this approach, at the start of execution, the root node is initialized with several random models. These can be randomly generated models, or models generated from robot motor babbling. If this pre-initialization step is applied, there is no delay in the formation of the first node, and the algorithm performance becomes robust to this type of presentation order, and identical to the results described in Section 4.2.1.

5. Conclusions

This paper develops a novel approach towards on-line, long-term incremental learning and hierarchical organization of whole body motion primitives. The learned motions are aggregates of the observed motions, which have been autonomously clustered during observation. The appropriate level of accuracy required for each motion model is determined based on the similarity of the motions to be distinguished, such that a larger model is only used in dense regions of the knowledge base. A sequential training approach is introduced, to allow incremental training of the Markov chains, as the model accuracy requirements increase. The proposed algorithm achieves comparable results to the exact training algorithm, while significantly reducing the training time, and allowing existing model knowledge to be reused.

The clustered motions are organized into a hierarchical tree structure, where nodes closer to the root represent broad motion descriptors, and leaf nodes represent more specific motion patterns. The tree structure and level of specialization will be based on the history of motions observed by the robot. The resulting knowledge structure is easily searchable for recognition tasks, and can also be utilized to generate the learned robot motions.

Acknowledgments

This work is supported by the Japanese Society for the Promotion of Science grant 18.06754 and Category S Grant-in-Aid for Scientific Research 15100002.

The authors also wish to thank the anonymous reviewers for the thorough and insightful review comments. Their suggestions significantly improved the quality and readability of the paper.

Appendix: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org.

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Video of animations of sample motions from the database and motions generated by the sequentially trained FHMM models
2	Data	Zip file containing the test data used in the experiments. Authors: Hiroaki Tanie, Koji Tatani and Yoshihiko Nakaniura. Please see the file readme.txt in the zip file for a description of the file contents and format

References

Bennewitz, M., Burgard, W., Cielniak, G. and Thrun, S. (2005). Learning motion patterns of people for compliant robot motion. *International Journal of Robotics Research*, **24**(1): 31–48.

Bentivegna, D. C., Atkeson, C. G. and Cheng, G. (2006). Learning similar tasks from observation and practice. *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 2677–2683.

Bernardin, K., Ogawara, K., Ikeuchi, K. and Dillmann, R. (2005). A sensor fusion approach for recognizing continuous human grasping sequences using hidden Markov models. *IEEE Transactions on Robotics*, **21**(1): 47–57.

Betkowska, A., Shinoda, K. and Furui, S. (2006). FHMM for robust speech recognition in home environment. *Proceedings of th Symposium on Large Scale Knowledge Resources*, pp. 129–132.

Billard, A., Calinon, S. and Guenter, F. (2006). Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, **54**: 370–384.

Breazeal, C. and Scassellati, B. (2002). Robots that imitate humans. *Trends in Cognitive Sciences*, **6**(11): 481–487.

Calinon, S. and Billard, A. (2007a). Active teaching in robot programming by demonstration. *Proceedings of the IEEE International Conference on Robot and Human Interactive Communication*, pp. 702–707.

Calinon, S. and Billard, A. (2007b). Incremental learning of gestures by imitation in a humanoid robot. *Proceedings of the ACM/IEEE International Conference on Human–Robot Interaction*, pp. 255–262.

Calinon, S., Guenter, F. and Billard, A. (2007). On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, **37**(2): 286–298.

Dillmann, R. (2004). Teaching and learning of robot tasks via observation of human performance. *Journal of Robotics and Autonomous Systems*, **47**: 109–116.

Dillmann, R., Rogalla, O., Ehrenmann, M., Zollner, R. and Bordegoni, M. (1999). Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm. *Proceedings of the International Symposium on Robotics Research*, pp. 229–238.

Dixon, K. R., Dolan, J. M. and Khosla, P. K. (2004). Predictive robot programming: theoretical and experimental analysis. *International Journal of Robotics Research*, **23**(9): 955–973.

Ekvall, S., Aarno, D. and Kragic, D. (2006). Online task recognition and real-time adaptive assistance for computer-aided machine control. *IEEE Transactions on Robotics*, **22**(5): 1029–1033.

Ezaki, H. (2000). Understanding actions of others through self-action components. *Master's Thesis*, University of Tokyo (in Japanese).

Ghahramani, Z. and Jordan, M. I. (1997). Factorial hidden Markov models. *Machine Learning*, **29**: 245–273.

Ho, M. A. T., Yamada, Y. and Umetani, Y. (2005). An adaptive visual attentive tracker for human communicational behaviors using HMM-based TD learning with new state distinction cpapbility. *IEEE Transactions on Robotics*, **21**(3): 497–504.

Iba, S., Paredis, C. J. J. and Khosla, P. K. (2005). Interactive multi-modal robot programming. *International Journal of Robotics Research*, **24**(1): 83–104.

Inamura, T., Toshima, I., Tanie, H. and Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *International Journal of Robotics Research*, **23**(4–5): 363–377.

Jacobs, R. A., Jiang, W. and Tanner, M. A. (2002). Factorial hidden Markov models and the generalized backfitting algorithm. *Neural Computation*, **14**: 2415–2437.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, **31**(3): 264–323.

Kadone, H. and Nakamura, Y. (2005). Symbolic memory for humanoid robots using hierarchical bifurcations of attractors in nonmonotonic neural networks. *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 2900–2905.

Kadone, H. and Nakamura, Y. (2006). Segmentation, memorization, recognition and abstraction of humanoid motions based on correlations and associative memory. *Proceedings of the IEEE–RAS International Conference on Humanoid Robots*, pp. 1–6.

Kragic, D., Marayong, P., Li, M., Okamura, A. M. and Hager, G. D. (2005). Human–machine collaborative systems for microsurgical applications. *International Journal of Robotics Research*, **24**(9): 731–742.

Kulić, D., Takano, W. and Nakamura, Y. (2007a). Incremental on-line hierarchical clustering of whole body motion patterns. *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication*, pp. 1016–1021.

Kulić, D., Takano, W. and Nakamura, Y. (2007b). Representability of human motions by factorial hidden Markov models. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 2388–2393.

Kuniyoshi, Y., Inaba, M. and Inoue, H. (1989). Teaching by showing: generating robot programs by visual observation of human performance. *Proceedings of the International Symposium on Industrial Robots*, pp. 119–126.

Kuniyoshi, Y. and Inoue, H. (1994). Learning by watching: extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, **10**(6): 799–822.

Lee, D., Kulić, D., and Nakamura, Y. (2008). Missing motion data recovery using factorial hidden Markov models. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1722–1728, 2008.

Lee, D. and Nakamura, Y. (2005). Mimesis from partial observations. *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 1911–1916.

Liu, S. and Asada, H. (1992). Transferring manipulative skills to robots: representation and acquisition of tool manipulative skills using a process dynamics model. *Journal of Dynamic Systems, Measurement and Control*, **114**(2): 220–228.

Murphy, K. P. (1999). Bayesian map learning in dynamic environments. *Advances in Neural Information Processing Systems*. Cambridge, MA, MIT Press.

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S. and Kawato, M. (2004). Learning from demonstration ADN adaptation of biped locomotion. *Robotics and Autonomous Systems*, **47**: 79–91.

Nicolescu, M. N. and Matarić, M. J. (2005). Task learning through imitation and human–robot interaction. *Imitation and social Learning in robots, Humans and Animals: Behavioral, Social and Communicative Dimensions*, Dautenhahn, K. and Nehaniv, C. (eds). Cambridge, Cambridge University Press.

Ogata, T., Sugano, S. and Tani, J. (2005). Open-end human–robot interaction from the dynamical systems perspective: mutual adaptation and incremental learning. *Advanced Robotics*, **19**: 651–670.

Ohkawa, K. and Nakamura, Y. (2005). Motion recognition by hierarchization and combination of statistical motion models. *Proceedings of the JSME Conference on Robotics and Mechatronics*, 1P1–S–009.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2): 257–286.

Rizzolatti, G. and Craighero, L. (2004). The mirror-neuron system. *Annual Reviews of Neuroscience*, **27**: 169–192.

Rizzolatti, G., Fogassi, L. and Gallese, V. (2001). Neurophysical mechanisms underlying the understanding and imitation of action. *Nature Reviews: Neuroscience*, **2**: 661–670.

Schaal, S., Ijspeert, A. and Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical*

Transactions of the Royal Society of London B: Biological Sciences, **358**: 537 – 547.

Startner, T. and Pentland, A. (1995). Visual recognition of american sign language using hidden Markov models. *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 189–194.

Takano, W. (2006). Stochastic segmentation, proto-symbol coding and clustering of motion patterns and their application to signifiant communication between man and humanoid robot. *Ph.D. Thesis*, University of Tokyo.

Takano, W. and Nakamura, Y. (2006). Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. *Proceedings of the IEEE–RAS International Conference on Humanoid Robots*, pp. 425–431.

Takano, W., Yamane, K., Sugihara, T., Yamamoto, K. and Nakamura, Y. (2006). Primitive communication based on motion recognition and generation with hierarchical mimesis model. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3602–3608.

Taylor, G. W., Hinton, G. E. and Roweis, S. (2006). Modeling human motion using binary latent variables. *Proceedings of the Conference on Neural Information Processing Systems*, pp. 1345–1352.

Yang, J., Xu, Y. and Chen, C. S. (1997). Human action learning via hidden Markov model. *IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans*, **27**(1): 34–44.