Incremental on-line hierarchical clustering of whole body motion patterns

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

***Abstract*— This paper describes a novel algorithm for autonomous and incremental learning of motion pattern primitives by observation of human motion. Human motion patterns are abstracted into a Hidden Markov Model representation, which can be used for both subsequent motion recognition and generation, analogous to the *mirror neuron* hypothesis in primates. As new motion patterns are observed, they are incrementally grouped together using hierarchical agglomerative clustering based on their relative distance in the HMM 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.**

I. 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 [3], [13]. 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. In our research, the goal is to implement continuous learning over the entire lifespan of the robot, where demonstrated actions are observed and classified autonomously, and learned 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 must be determined autonomously by the robot, as it is observing the motions. Due 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. 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 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 [1] and feedback from the teacher [9], which may be more effective than repeated observation alone.

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.

Our motion model is inspired by the mirror neuron system, found in humans and other primates [12]. 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 *observer's* motor representation of the same action. A motion is understood by causing the motor system of the observer to "resonate". This hypothesis is supported by both experiments with monkeys and humans, where observation of a motor action generates a neural response in the motor areas of the brain corresponding to the observed action. The neurons which respond in this manner have been named *mirror neurons*. In previous research [5], [14], humanoid motion primitives have been encoded using Hidden Markov Models, 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 behavior acquisition and learning of motion primitives, and the automated organization of the acquired behaviors.

A. Related Work

Breazeal and Scasellati [3] and Schaal et al. [13] provide reviews 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.

For example, Billard et al. [2] use HMM models for motion recognition and generation. The Bayesian Information Criterion (BIC) is used to select the optimal number of states for the HMM. 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.

Ogata et al. [10] develop a connectionist architecture suitable for long term, incremental learning. However, in their implementation, the robot learns only one task, and no hierarchical organization of knowledge takes place.

Kadone and Nakamura [7], [8] describe a system for automated segmentation, memorization, recognition and abstraction of human motions based on associative neural networks 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 [15] develop a system for automated segmentation, recognition and generation of human motions based on Hidden Markov Models. 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.

In this paper, an HMM representation is used to abstract motion patterns as they are perceived. Individual motion patterns are then clustered in an incremental fashion, based on intra HMM distances. The formed clusters are then used to form group HMMs, which can be used for motion generation. Section II describes the Hidden Markov Model encoding of each motion primitive. The incremental behavior learning and hierarchy formation algorithm is described in Section III. Section IV describes the motion generation algorithm used to generate motion commands based on the stored HMM motion primitives. Experimental results on a motion capture data set are described in Section V; concluding remarks are presented in Section VI.

II. Motion Primitive Encoding

Each motion primitive p is encoded as a stochastic Hidden Markov Model (HMM) $\lambda_p(\pi, A, B)$, where π is the initial state probability, $A_{N\times N}$ is the $N \times N$ state transition matrix, and B is the output observation model. N represents the number of states in the model. For representing non-periodic human motion primitives, a left-to-right HMM is used. In addition, the state transition matrix is constrained such that no backwards state transitions are allowed, and state transitions can occur only between consecutive nodes. The output observation vector is composed of continuous observations. Either joint angles, or cartesian positions of key body locations (such as the wrist, elbow, head, knee, etc.) can be used as the observation vector. The output observation model B is a finite mixture of Gaussians. The number of Gaussians mixtures is denoted by M. To reduce the number of training parameters, the covariance matrix $\mathbf{\Sigma}_{jm}$ is constrained to be diagonal.

When a motion is encoded into an HMM, it is important to select the best model representation, by selecting the appropriate number of states N and number of Gaussian mixtures M for the motion. The selection of model is performed autonomously based on the observed motion sequence, as will be described in the following section.

Once the model is selected, the model parameters $a_{ij}, c_{jm}, \boldsymbol{\mu}_{jm}, \mathbf{\Sigma}_{jm}$ must be trained, based on the observation data. The training procedure used is the Baum-Welch algorithm, which is a type of Expectation-Minimization algorithm [11].

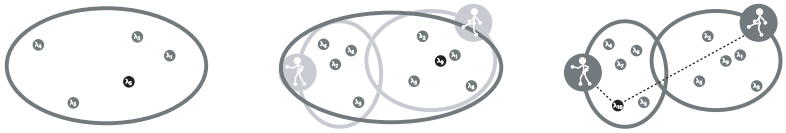
Once the model is trained, we can use the *forward procedure* [11] to determine the likelihood $P(\mathbf{O}_{new}|\lambda)$ that a new observation sequence $\mathbf{O}_{new}$ was generated by the model λ.

III. 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 our 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.

The algorithm initially begins with one behavior group (the root node). Each time a motion is observed from the teacher, it is encoded into an HMM, using all available sensor information. Behaviors are encoded using the continuous Bakis model (left to right) HMMs [14]. The HMM-encoded motion is then compared to existing behavior groups via a tree search algorithm, using the symmetric HMM distance measure [11], and placed into the closest group. Each time a group is modified, a hierarchical agglomerative clustering algorithm [6] is performed within the exemplars of the group. If a sub cluster starts to form, a child group is formed with the sub cluster data. The raw data of the HMMs forming the sub cluster is then used to generate a single group HMM, 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. A schematic representation of the algorithm is shown in Figure 1. The algorithm pseudocode is shown in Figure 2.

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



(a) (b) (c)

Fig. 1. Schematic Illustration of the Segmenting Algorithm. (a) initial state, when only one group is present; (b) clusters form 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

Step1 Encode observation sequence O_i into an HMM λ_i
Step2 Calculate the distance between λ_i and each existing behavior group HMM λ_{Gj}
Step3 Place λ_i into the closest group G_c
Step4 Cluster all HMMs within G_c
Step5 If a cluster forms, form a new group G_n, containing the HMMs of the cluster
Step6 Using the observations sequences from the HMMs in G_n, form the group HMM λ_{Gn}

Fig. 2. Segmenting Algorithm Pseudocode

specialized in those areas of the behavior space where the most data has been observed.

A. Step1 - Observation Sequence Encoding

In the first step, the newly acquired observation sequence is encoded into an HMM. Once the observation vector is generated, the HMM parameters, such as the number of states and the number of gaussian mixtures must be selected. The Akaike Information Criterion (AIC) is used to select the best fitting model [4]. The AIC is given by Equation 1:

$$AIC = -2\mathcal{L} + 2n_p + \frac{2n_p(n_p+1)}{N - n_p - 1} \quad (1)$$

$\mathcal{L}$ denotes the log likelihood that the given model produced the data, and n_p denotes the total number of independent parameters of the model. The AIC is based on Kullback-Liebler information, and can be thought of as an entropy measure [4]. For small data sets, a correction factor must be applied to the AIC to account for second order effects [4]. The last term in Equation 1 is the correction factor, where N indicates the total number of data points. The AIC formulates a tradeoff between the "goodness of fit" of the candidate model and the number of model parameters.

The model selection phase can be quite time consuming, as each model candidate must be trained in order to be evaluated by the AIC. To narrow the model search space, in the initial encoding, the output is initially modeled as a single Gaussian probability density function, and the maximum number of HMM states is limited to 10% of the number of observations.

B. Step 2 - HMM 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 [11] by Equation 2.

$$D(\lambda_1, \lambda_2) = \frac{1}{T}[logP(O^{(2)}|\lambda_1) - logP(O^{(2)}|\lambda_2)] \quad (2)$$

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:

$$D_s = \frac{D(\lambda_1, \lambda_2) + D(\lambda_2, \lambda_1)}{2} \quad (3)$$

Note that Equation 3 is the pseudo-distance and not the distance in the strict sense since it does not satisfy the triangular equation. 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.

$$D_{thresh} = K_{maxGD} D^G_{max} \quad (4)$$

D_{thresh} is the distance threshold at which a new observation sequence is considered for inclusion to a node, K_{maxGD} 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.

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.

C. Steps 4 and 5 - Clustering and New Group Formation

When a new observation sequence is added to a cluster, a clustering procedure is invoked on that cluster, to determine

if a subcluster may be formed. The complete link hierarchical clustering algorithm is used to generate the data clusters within a group [6]. 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: 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. To calculate the maximum distance measure, the average and standard deviation of the inter motion distances in the cluster is calculated. The distance cutoff is then calculated as a function of the distribution function, as shown in Equation 5.

$$D_{cutoff} = \mu - K_{cutoff}\sigma \tag{5}$$

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

D. Step 6 - New Behavior Instantiation

If a new cluster is generated in Step 5, a new group is instantiated, containing the inter-cluster elements. A new group HMM is trained using the raw observation sequences from all the group elements. This HMM is subsequently used by the robot to generate behaviors. The group HMM replaces the individual observations in the parent node.

If one of the group elements allocated to the new cluster is already a group HMM, the generated motion sequence based on that HMM will be used for the training. The generation algorithm is described in Section V. In this case, a modified form of the re-estimation formulas for multiple observation sequences [11] is used. The algorithm is modified by over-weighting the group HMMs, in order to account for the fact that there are multiple observation sequences stored in the generated HMM, and therefore more weight should be given to the group HMM, as compared to the individual observation sequence HMM. The modified re-estimation formula for the state transition matrix elements is given by Equation 6.

$$\bar{a_{ij}} = \frac{\sum_{k=1}^{K} \frac{W_k}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(O_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{W_k}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{6}$$

where $\bar{a_{ij}}$ is the new estimate of the i, j element of the state transition matrix, K is the number of group elements being used to train the HMM, W_k is the number of examples contained in each element, and P_k is the probability that the generated (or, in the case of an individual observation, observed) data sequence was generated by the current HMM model. T_k is the number of observations in the sequence $O^{(k)}$, and α^k and β^k are the forward and backward variables for the sequence k. The weighting factor W_k accounts for the multiple sequences stored in the group HMM. A similar modification is made to the re-estimation formulas for the output observation model parameters. Weighting is also used to account for group HMMs during the new group formation threshold calculation.

1) Superseding: A special case that may occur during new cluster formation is that group HMMs in a parent cluster may become targeted for inclusion in a newly formed cluster. Once a new subcluster is formed, the behaviors forming the subcluster are removed, and replaced with a single, group HMM. Thereafter, this group HMM is treated exactly the same as a single behavior HMM, except that it is overweighted during cluster group HMM training. Therefore, as more behaviors get added to the cluster, occasionally the group HMM may be included in a new subcluster. In this case, the tree structure is also updated to insert the newly formed subcluster as the parent of the group HMM which has been placed in the cluster.

E. 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. 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 HMMs 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.

IV. Motion Generation

Once a cluster node has been formed, the group HMM for the node constitutes the abstraction of the motion primitive. To generate a motion trajectory for the robot from the group HMM, the Monte-Carlo like averaging method of Takano [14] is used. In this approach, a representative state sequence if first generated by sampling from the state distribution. The output sequence is then generated by sampling from each output observation distribution. This two-stage sampling process is then averaged over many trials, to average out the noise due to potentially large variances in the output observation vector. The resulting reference trajectory is then passed to a low level controller, to ensure that dynamic and stability constraints are satisfied.

V. SIMULATIONS

The algorithm has been tested on a data containing a series of 9 different human movement observation sequences obtained through a motion capture system [7]. The data set contains joint angle data for a 20 degree of freedom 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 3 shows an example of a walking motion from the dataset.

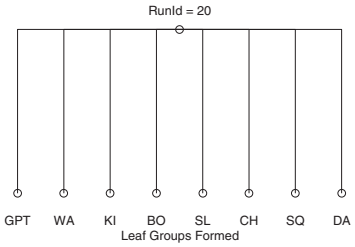


Fig. 4. Sample Segmentation Result, $K_{cutoff} = 1.2$

The 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. Sample segmentation results are shown in Figs. 4, 5 and 6. An example of a motion generated by the walk cluster is shown in Figure 7. Note that the actual order of node formation will vary depending on the motion presentation order.

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

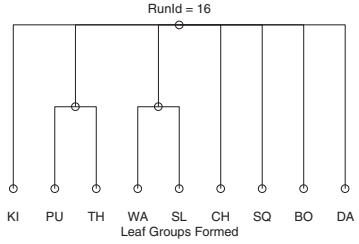


Fig. 5. Sample Segmentation Result, $K_{cuttoff} = 0.9$

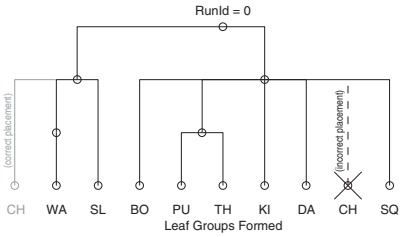


Fig. 6. Sample Segmentation Result, $K_{cuttoff} = 0.9$

shown in Figure 4, while low values of K_{cutoff} result in a deep tree structure, as shown in Figure 5. As the cluster formation parameter is relaxed, deeper trees tend to be formed. However, the resulting tree structure tends to be dependant on the presentation order. In the case of a high cutoff value (see Figure 4), 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 to form (indicated by -1 in Figure 4), 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 GPT in Figure 4). The generated motion resulting from that subcluster is shown in Figure 8. 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 dependant on presentation order. The similarity level at which nodes will form is highly dependent on presentation order. Figures 5 and 6 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. Even though the cutoff level was the same for both experiments, the similarity level of the nodes formed differed, based on the presentation order. The result in Figure 5 is consistent with global clustering, while in the result shown in 6, 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.

VI. CONCLUSIONS

This paper develops a novel algorithm for the incremental learning and hierarchical organization of whole body motion primitives encoded into Hidden Markov Models. The learned

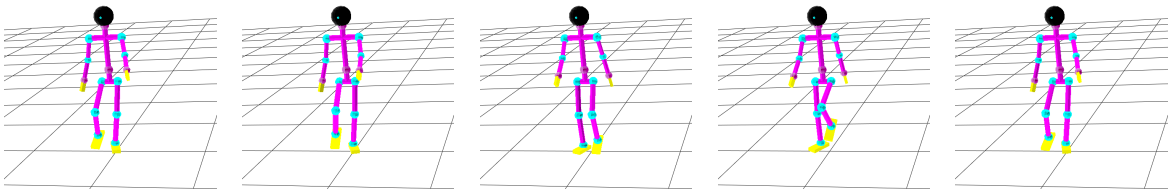


Fig. 3. Sample Walking Motion

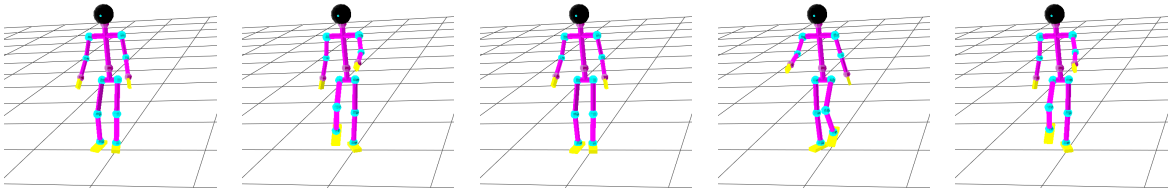


Fig. 7. Generated Walking Motion

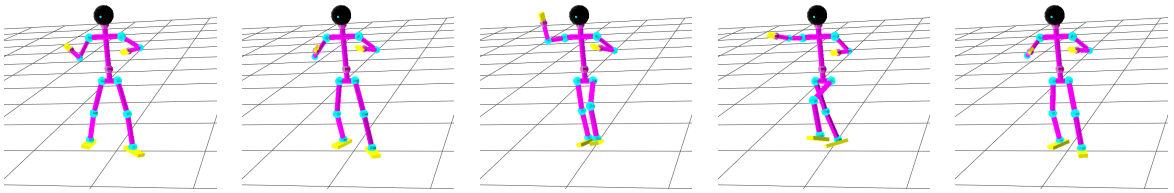


Fig. 8. Generated Hybrid Punch/Throw Motion

motion is an aggregate of the clustered motions, represented by the group HMM. A weighted training procedure is developed for handling group HMMs, such that the training and behavior generation is weighted by the number of observations represented by the group HMM. The clustered motions are organized in 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 developed algorithm was tested on a motion pattern database consisting of human motion patterns. The experiments showed that leaf nodes of the resulting tree structure represent the correct segmentation, regardless of the presentation order.

Acknowledgment

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.

References

[1] D. C. Bentivegna, C. G. Atkeson, and G. Cheng. Learning similar tasks from observation and practice. In *International Conference on Intelligent Robots and Systems*, pages 2677–2683, 2006.

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

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

[4] K. P. Burnham and D. R. Anderson. Multimodel inference: Understanding aic and bic in model selection. *Sociological Methods and Research*, 33(2):261–304, 2004.

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

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

[7] H. Kadone and Y. Nakamura. Symbolic memory for humanoid robots using hierarchical bifurcations of attractors in nonmonotonic neural networks. In *International Conference on Intelligent Robots and Systems*, pages 2900–2905, 2005.

[8] H. Kadone and Y. Nakamura. Segmentation, memorization, recognition and abstraction of humanoid motions based on correlations and associative memory. In *IEEE-RAS International Conference on Humanoid Robots*, pages 1–6, 2006.

[9] M. N. Nicolescu and M. J. Matarić. Task learning through imitation and human-robot interaction. In K. Dautenhahn and C. Nehaniv, editors, *Imitation and social learning in robots, humans and animals: behavioral, social and communicative dimensions*. Cambridge University Press, 2005.

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

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

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

[13] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358:537 – 547, 2003.

[14] W. Takano. *Stochastic segmentation, proto-symbol coding and clustering of motion patterns and their application to signifiant communication between man and humanoid robot*. PhD thesis, University of Tokyo, 2006.

[15] W. Takano and Y. Nakamura. Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. In *IEEE-RAS International Conference on Humanoid Robots*, pages 425–431, 2006.