2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Combining Automated On-line Segmentation and Incremental Clustering for Whole Body Motions

Dana Kulić, Wataru Takano and Yoshihiko Nakamura

Abstract— **This paper describes a novel approach for incremental learning of human motion pattern primitives through on-line observation of human motion. The observed motion time series data stream is first stochastically segmented into potential motion primitive segments, based on the assumption that data belonging to the same motion primitive will have the same underlying distribution. The motion segments are then abstracted into a stochastic model representation, and automatically clustered and organized. As new motion patterns are observed, they are incrementally grouped together based on their relative distance in the model space. The resulting representation of the knowledge domain is a tree structure, with specialized motions at the tree leaves, and generalized motions closer to the root. The tree leaves, which represent the most specialized learned motion primitives, are then passed back to the segmentation algorithm, so that as the number of known motion primitives increases, the accuracy of the segmentation can also be improved. The combined algorithm is tested on a sequence of continuous human motion data obtained through motion capture, and demonstrates the performance of the proposed approach.**

I. Introduction

Learning by observing and imitating humans is an attractive proposition for humanoid robots, taking advantage of the similarity in body structure between humanoids and humans. Many algorithms have been proposed in the literature [1], [2]. However, most of the approaches thus far consider off-line learning, where the data is collected, segmented and sorted into the motion groups to be learned a priori. 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 segment and classify demonstrated actions on-line during co-location and possible interaction with the (human) teacher.

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. Since data is being processed autonomously, in an on-line system, later stages of the process must be robust to errors in the preceding steps. We propose a combined approach for on-line segmentation and clustering of whole body human motion patterns. The observed motion time series data stream is first stochastically segmented into potential motion primitive segments, based on the assumption that data belonging to the same motion primitive will have the same underlying distribution. The segmented motions are then passed to an incremental clustering algorithm which forms a tree representation of the learned motions, and abstracts each motion type into a generative model. The learned motion primitives are then passed back to the segmentation algorithm, so that, as the motion is viewed repeatedly over time, both the segmentation performance and the model of the motion primitive improves.

The authors are with the Department of Mechano-Informatics, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, 113-8656 Tokyo, Japan {dana,takano,nakamura}@ynl.t.u-tokyo.ac.jp

A. Related Work

The first requirement of autonomous motion learning is the segmentation of the incoming data stream into appropriate segments, representing potential motion primitives. Existing data segmentation algorithms can be divided into two broad categories: unsupervised algorithms which require no a-priori knowledge of the motion data to be segmented, and algorithms which make use of a-priori specified motion primitives to perform the segmentation.

In the first category, some assumption is made about the underlying structure of the data at a segmentation point. For example, several algorithms have been developed for segmenting motions based on the velocity properties of the joint angle vector [3], [4], [5]. In Pomplun and Matarić [3], a segment is recognized when the root mean square (RMS) value of the joint velocities falls below a certain threshold. In this case, the assumption is that there will be a pause in the motion between motion primitives. In Fod et al. [4], it is assumed that there is a change in the direction of movement accompanying a change between motion primitives. Therefore, a segmentation point is recognized when a Zero Velocity Crossing (ZVC) is detected in the joint angle data, in a sufficient number of dimensions. However, with all the velocity based approaches, it becomes more difficult to tune the algorithm as the number of joints increases.

Koenig and Matarić [6] develop a segmentation algorithm based on the variance of the feature data. The algorithm searches for a set of segment points which minimize a cost function of the data variance. In a related approach, Kohlmorgen and Lemm [7] describe a system for automatic on-line segmentation of time series data, based on the assumption that data from the same motion primitive will belong to the same underlying distribution. The incoming data is described as a series of probability density functions, which are formulated as the states of a Hidden Markov Model (HMM), and a minimum cost path is found among the states using an accelerated version of the Viterbi algorithm.

Janus and Nakamura [8], [9] apply this approach to human motion capture data.

In the second class of segmentation algorithms, motion primitives are specified by the designer a-priori, and segmentation is based on the comparison between the known motions and the incoming data. For example, in Takano and Nakamura [10], [11] the known motion primitives are encoded via short HMMs. Segmentation points are then decided based on the error between the motion data predicted by the HMM and the actual observed motion data. If the error increases above a certain threshold, a segment point is declared.

Once the data is segmented into motion primitive exemplars, the motion primitives must be clustered and abstracted. Breazeal and Scasellati [1] and Schaal et al. [2] 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 and clustered a-priori, and that the model training takes place off-line. For example, Billard et al. [12] 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.

Kadone and Nakamura [13], [14] describe a system for automated segmentation, memorization, recognition and abstraction of human motions based on associative neural networks with non-monotonic sigmoid functions. However, the abstracted motion representation can only be used for subsequent motion recognition, and cannot be used for motion generation.

Kulić et al. [15], [16] have been developing algorithms for incremental learning of motion pattern primitives through long-term observation of human motion. Human motion patterns are abstracted into a stochastic model representation, which can be used for both subsequent motion recognition and generation. The model size is adaptable based on the discrimination requirements in the associated region of the current knowledge base. As new motion patterns are observed, they are incrementally grouped together based on their relative distance in the model space. The resulting representation of the knowledge domain is a tree structure, with specialized motions at the tree leaves, and generalized motions closer to the root.

B. Proposed Approach

The aim of our research is to develop robots which can learn motion primitives on-line while observing and interacting with a human partner. Therefore we seek a segmentation method which can begin processing the incoming data in an unsupervised manner, similar to the approach of Janus and Nakamura [8], but which can improve its performance over time, as more motions become clustered and abstracted as motion primitives. The learned motion primitives can be used to improve the segmentation performance.

We use a modified version of the Kohlmorgen and Lemm [7] algorithm for unsupervised segmentation of on-line human motion data, and then input the extracted segments into an automated clustering and hierarchical organization algorithm [15], [16]. The segmentation algorithm uses a Hidden Markov Model to represent the incoming data sequence, where each model state represents the probability density estimate over a window of the data. The segmentation is implemented by finding the optimum state sequence over the developed model. The clustering algorithm uses a variable structure Hidden Markov Model based representation 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 model of each abstracted motion primitive, which can be used for subsequent motion generation. As the observed motion primitives become known, the segmentation algorithm is modified to include permanent states, representing the known motion primitives. The permanent states are then used together with the temporary states generated from the current observation window to generate the optimum state sequence which represents the segmentation result. Therefore as more motion primitives become known, they are also used to improve the segmentation results. This paper presents the combined segmentation and clustering approach, which can be used to autonomously extract motion primitives from continuous on-line observation of a human demonstrator. The proposed approach is robust to initial errors in segmentation, and quickly abstracts the motion segments, such that both the segmentation and motion representation components together converge to improve performance over time. Section 2 summarizes the segmentation algorithm, Section 3 overviews the clustering method, while Section 4 describes the combined approach. In Section 5, the results of experiments verifying the algorithm on a continuous stream of human motion capture data is reported. Section 6 concludes the paper.

II. Probabilistic Segmentation with Scaffolding States

The Kohlmorgen and Lemm [7] segmentation algorithm is based on the assumption that data belonging to the same motion primitive will have the same underlying probability distribution. The incoming data stream is first embedded into a higher-dimensional space,

$$\vec{x_t} = (\vec{y_t}, \vec{y_{t-1}}, \ldots, \vec{y_{t-(m-1)\tau}}), \quad (1)$$

where $\vec{y_1}, \vec{y_2}, \vec{y_3}, \ldots$ is an incoming data stream to be analyzed, m is the embedding dimension and τ is the delay parameter. Next, the density distribution of the embedded data is estimated over a sliding window of length W, via a standard density estimator with multivariate Gaussian kernels, centered on the data points in the window $\{\vec{x_{t-w}}\}_{w=0}^{W-1}$.

$$p_t(\mathbf{x}) = \frac{1}{W} \sum_{w=0}^{W-1} \frac{1}{(2\pi\sigma^2)^{d/2}} exp(-\frac{(\mathbf{x} - \vec{x_{t-w}})^2}{2\sigma^2}), \quad (2)$$

where σ is a smoothing parameter calculated proportional to the mean distance between each $\vec{x_t}$ and its d nearest neighbors.

As more data are observed, the distance between successive data windows can be calculated based on the integrated square error between two probability density functions. This distance can be calculated analytically in the case of mixtures of Gaussian density functions:

$$\begin{aligned} d(p_{t1}(\mathbf{x}), p_{t2}(\mathbf{x})) = & \frac{1}{W^2(4\pi\sigma^2)^{d/2}} \\ & \sum_{w,v=0}^{W-1} [exp(-\frac{(\vec{x_{t1-w}} - \vec{x_{t1-v}})^2}{4\sigma^2}) \\ & - 2exp(-\frac{(\vec{x_{t1-w}} - \vec{x_{t2-v}})^2}{4\sigma^2}) \\ & + exp(-\frac{(\vec{x_{t2-w}} - \vec{x_{t2-v}})^2}{4\sigma^2}). \end{aligned} \quad (3)$$

The analysis is carried out by defining a Hidden Markov Model over a set S of sliding windows. Each window corresponds to a state of the HMM. For each state, the observation probability distribution is defined as:

$$p(p_t(\mathbf{x})|s) = \frac{1}{\sqrt{2\pi}\varsigma} exp(-\frac{d(p_s(\mathbf{x}), p_t(\mathbf{x}))}{2\varsigma^2}), \quad (4)$$

where $p(p_t(\mathbf{x})|s)$ is the probability of observing the window represented by $p_t(\mathbf{x})$ in state s.

The basic algorithm is modified to take advantage of known motions. Known motions could be provided manually from the user, or through the use of an automated clustering technique [15], [16].

The known motion is first encoded into a window representation by embedding the data in the same manner as the input data, as described in Equation 1. Since exemplar motions can be of different durations, the exemplar data can span over multiple windows. Once the exemplar data has been embedded, the distance between the known motion(s) and the current temporary states is calculated as the minimum over all the distance measures between the temporary state and each window in the known motion.

$$D_k(s_t, s_p) = \max_{i=1:n_{sp}} D(p_t, p_{pi}) \quad (5)$$

$D_k(s_t, s_p)$ is the distance between temporary state s_t and known motion s_p,n_{sp} is the number of windows for the known motion described by s_p, and $D(p_t, p_{pi})$ is the distance between two windows as shown in Equation 3. Temporary states for which the distance between the state and one of the known motions is small are identified as known motions, termed *permanent states*.

The structure of the HMM is then modified to bias the model to prefer permanent states over temporary states. The state transition matrix A is modified such that transitions to a permanent state are more likely then transitions to a temporary state. The modified state transition matrix is given by Equation 6.

$$a_{ij} = \begin{cases} \frac{k}{C} & \text{if } i = j; \\ \frac{1}{C} & \text{if } i \neq j \text{ and } i \in \mathcal{S}_t; \\ \frac{K_s}{C} & \text{if } i \neq j \text{ and } i \in \mathcal{S}_p. \end{cases} \quad (6)$$

where K_s is a factor favoring a transition to a permanent state over a transition to a temporary state, $1 < K_s < k$, and C is a normalizing constant ensuring that each row of the state transition matrix sums to 1.

In addition, known motion patterns are used to modify the variance of the observation function of the known states and extract information about which degree of freedom is most active during the motion. The relative activity of each degree of freedom is calculated as follows:

$$j_i^{act} = \frac{1}{T_p} \sum_{t=0}^{T_p} (j_i(t) - \bar{j_i})^2 \quad (7)$$

where j_i^{act} is the activity of each degree of freedom, T_p is the length of the known motion sequence, $j_i(t)$ is the value of DoF i at time t, and $\bar{j_i}$ is the average value of the DoF over the entire known motion sequence.

This activity value is then used to weigh the distance computation when a permanent state is active.

$$\begin{aligned} D_w(p_{t1}(\mathbf{x}), p_{t2}(\mathbf{x})) = & \frac{1}{L^2(4\pi\sigma_k^2)^{d/2}} \\ & \sum_{i,j=0}^{L-1} [exp(-\frac{W(\vec{x_{t1-i}} - \vec{x_{t1-j}})^2}{4\sigma_k^2}) \\ & - 2exp(-\frac{W(\vec{x_{t1-i}} - \vec{x_{t2-j}})^2}{4\sigma_k^2}) \\ & + exp(-\frac{W(\vec{x_{t2-i}} - \vec{x_{t2-j}})^2}{4\sigma_k^2}) \end{aligned} \quad (8)$$

where D_w is the weighted distance and W is the vector of weights, with each element proportional to the activity of the corresponding degree of freedom. σ_w is the combined variance computed over the entire known motion window.

The modified algorithm is outlined in Figure 1. In the algorithm, $C_t = 2\varsigma^2 \log(k)$ represents the transition cost of switching to a new temporary state state in the path, while $C_p = 2\varsigma^2 \log(k/K_s)$ represents the transition cost of switching to a permanent state in the path. $c_s(t)$ is the cost of following the optimum path and ending in state s at time t, and $o(t)$ is the optimum cost at time t.

To prevent the state list from growing to infinity as the number of data points observed increases, Kohlmorgen and Lemm [7] propose removing states following a segment away from that state. However, Janus [9] have found that this approach leads to over-segmenting, as the considered data range becomes too small (on the order of $5W$) and therefore the algorithm becomes more prone to local minima. Instead, Janus propose that the algorithm is run in batch-mode over a larger, fixed number of windows, and that windows be discarded in a FIFO manner. The Janus approach is adopted herein.

```
procedure OnLineViterbiScaffolded
    Initialize
    $c_0 \leftarrow o_0 \leftarrow d_{0,0} \leftarrow 0$
    if $s_p \in \mathcal{S}_p$ s.t. $(D(r, s_p) < D_t)$ then
        $r = s_p$
        $C_{sw} = C_p$
        $d_{r,t} = D_w(r, t)$
    else
        $r = s_t$
        $C_{sw} = C_t$
        $d_{r,t} = D(r, t)$
    end if
    for $t \leftarrow 0, T - 1$ do
        Find optimum path ending in new state r
        $c_r(t) \leftarrow d_{r,t} +$
        $\begin{cases} 0 & \text{, if } t = 0 \\ \min\left(c_r(t-1), o(t-1) + C_{sw}\right) & \text{, else} \end{cases}$
        if $c_r(t) < o(t)$ then
            $o(t) = c_r(t)$
        end if
    end for
    for $s \leftarrow 0, S$ do
        Compute min cost for each state at T
        if $s \in \mathcal{S}_t$ then
            $c_s(T) \leftarrow D(s, T) +$
            $\min\left(c_s(T-1), o(T-1) + C_t\right)$
        else
            $c_s(T) \leftarrow D_w(s, T) +$
            $\min\left(c_s(T-1), o(T-1) + C_p\right)$
        end if
    end for
    $o(T) \leftarrow \min_s\left(c_s(T)\right)$
end procedure
```

Fig. 1. Scaffolded Segmentation Algorithm Pseudocode

III. Incremental Behavior Learning

Once the incoming time series data has been segmented into potential primitives, each segment is sequentially passed to the clustering module. In the proposed clustering approach [15], [16], 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. Within each local area of the motion space, a standard clustering technique [17] is used to subdivide motion primitives. A Hidden Markov Model is used to abstract the observation sequences. The parameters of the model form the feature set of the data. These features are then used to define a distance measure between observation sequences, which is used for clustering.

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 and compared to existing behavior groups via a tree search algorithm, and placed into the closest group. Each time a group is modified, local clustering is performed within the exemplars of the group. If a a cluster with sufficiently similar data is found, a child group is formed with this data subset. 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 3, while a schematic of the incremental memory structure formation is shown in Fig. 2.

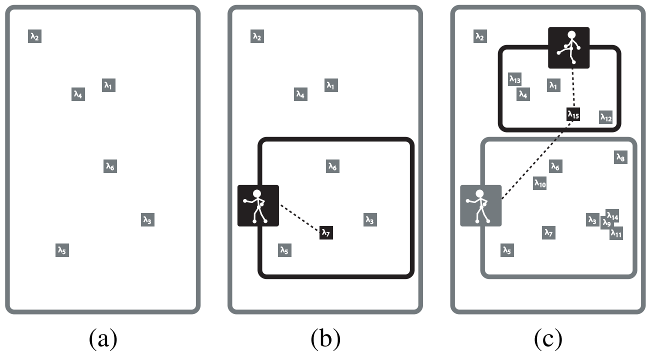

Fig. 2. Schematic Illustration of the Segmenting Algorithm. (a) initial state, when only one group is present; (b) a child group forms when enough similar examples are observed; (c) new observations are located into the closest group based on the distance between the new observation and the group model;

```
procedure IncrementalCluster
    Step1 Encode observation sequence $O_i$ into an HMM $\lambda_i$
    Step2 Search the behavior tree for the closest group $\lambda_{Gj}$ to the current observation model $\lambda_i$, based on the inter-model distance
    Step3 Place $\lambda_i$ into the closest group $G_c$
    Step4 Perform clustering on all the exemplar motions within $G_c$
    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
    Step6 Using the observations sequences of the new subgroup, form the group model $\lambda_{Gn}$
end procedure
```

Fig. 3. Segmenting Algorithm Pseudocode

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.

A. The Clustering Approach

Each newly acquired observation sequence is encoded into a Hidden Markov Model. It is then compared to existing groups (if any). Here, the distance between two models can be calculated [18] by Equation 9.

$$D(\lambda_1, \lambda_2) = \frac{1}{T}[logP(O^{(2)}|\lambda_1) - logP(O^{(2)}|\lambda_2)] \qquad (9)$$

where λ_1, λ_2 are two 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 distances is used to form a symmetric measure. 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 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. The new observation sequence is placed in the closest node, if the distance to the closest node is sufficiently small, based on the maximum node distance, otherwise it is placed in the parent node of the closest node.

$$D_{thresh} = K_{maxGD} D^{G}_{max} \tag{10}$$

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.

Once a new observation sequence is added to a group, 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 hierarchical tree structure within a group [17]. Clusters are formed based on two criteria: number of leaves in the subgroup, and the maximum proximity measure of the potential subgroup. 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:

$$D_{cutoff} = \mu - K_{cutoff}\sigma \tag{11}$$

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 among all the distances in the node.

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 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 [18] is used. The algorithm is modified by overweighting 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.

B. Motion Generation

Once a cluster node has been formed, the group model for the node constitutes the abstraction of the motion primitive. To generate a motion trajectory for the robot from the group model, the deterministic motion generation method is used [19]. In this method, at each time step, the state duration is first estimated from the state transition model, and the subsequent state is selected by a greedy policy. The output observation vector is then generated by a greedy policy on the output model. The resulting reference trajectory is then low-pass filtered and passed to a low level controller, to ensure that dynamic and stability constraints are satisfied.

IV. Combining Segmentation and Clustering

Due to the fact that the segmentation algorithm begins with no a-priori knowledge of the motion patterns, initially, imperfect segmentation results are received by the clustering algorithm. In addition, due to the fact that the data is being segmented on-line, the incoming data stream being analyzed by the segmentation algorithm may contain errors, such as errors due to marker mislabeling, or temporary marker occlusion. Our approach is based on the assumption that errors in segmentation will be quasi random, such that erroneous segments will be sufficiently dissimilar from correct segments, and sufficiently dissimilar from each other, as measured by Equation 9, so that they can be differentiated by the clustering algorithm.

To aid the performance of the clustering algorithm, especially during initialization, when there are no known motion primitives, the extracted motion segments are validated prior to being passed to the clustering algorithm. Segments which are too short, indicating a likely false positive error of the segmentation algorithm, or segments which are too long, indicating a likely false negative error, are excluded from consideration by the clustering algorithm.

Once a potential motion primitive has been abstracted by the clustering algorithm, the motion primitive is added to the segmentation module as a known motion (i.e., a set of permanent states) and used to scaffold further segmentation, as described in Section 2. To ensure that incorrectly abstracted motions do not get passed to the segmentation algorithm, a simple validation is performed on each abstracted motion. Only leaf nodes are used as known motions, as these nodes represent the most specific knowledge available to the clustering algorithm at a given time. In addition, only motions where the maximum group distance is lower than a specified threshold are admitted. The combined algorithm is presented in Figure 4.

V. Experiments

The combined algorithm was tested on a human motion data set consisting of 4 minutes of continuous whole body motion data of a single human subject. During the data sequence, the subject performs a variety of full body motions, including a walk in place motion, a squat motion, kicking and arm raising. The subject performs a total of 138 different motions. In some cases, there is a pause between

```
procedure CombinedSegmentationAndClustering
    while 1 do
        Observe Data Point
        call OnLineViterbiScaffolded
        if SegPoint then
            if IsValid(Segment) then
                call IncrementalCluster
                if IsValid(NewMotion) then
                    Add/Replace new motion as permanent state
                end if
            end if
        end if
    end while
end procedure
```

Fig. 4. Combined Segmentation and Clustering Algorithm Pseudocode

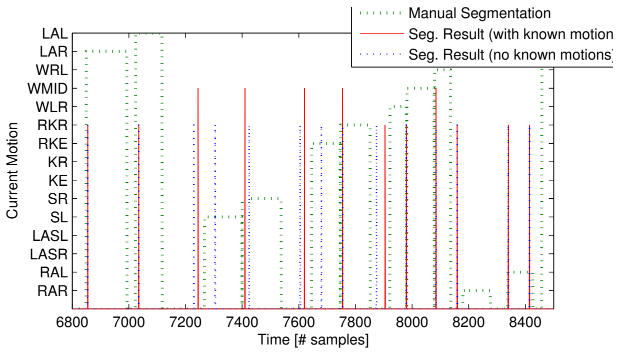

Fig. 5. Segmentation Result Detail for the Stand-alone Algorithm

motions, while other motions are fluidly connected. The motion capture system [20] captures the Cartesian position of markers located on the body (for example, shoulder, elbow, wrist, hip, knee, etc.) with a sampling rate of 10ms, and performs inverse kinematics computations to convert the data to joint angle positions in real time. A 26 degree of freedom humanoid model is used for the inverse kinematics computations (20 rotational joint angles and 6 degrees of freedom for the free body joint). The 20 element vector of rotational joint angle data is then down sampled to 30ms per sample, and passed to the segmentation algorithm.

In addition to errors due to incorrect state modeling, the segmentation algorithm can also fail due to errors in the incoming marker data, such as missing markers due to temporary occlusion, and mislabeled markers. The conversion from marker data to joint angle data via inverse kinematics reduces the occurrence of the first error [9], since the inverse kinematics acts as a kind of filter smoothing and incorporating raw marker data. However, if a large enough number of markers is missing, the inverse kinematics may temporarily fail to return correct data, resulting in segmentation errors. However, no post-processing was performed on the marker data, to ensure that the algorithm is robust in the presence of these errors.

First, the segmentation algorithm was tested in a standalone manner, with known motion primitives manually added. The basic algorithm achieves a correct segmentation rate of about 86%, but also suffers from a high rate of false positives (i.e., additional segmentation points inserted in the middle of a single motion primitive). A section of the segmented data stream is shown in Figure 5. In the figure, the dotted lines represent the manual segmentation points. The dashed line indicates the segmentation result generated by the basic algorithm, while the solid line indicates the segmentation result generated when 2 known motions (the kick and the squat motion) are added. The y axis indicates the action being performed, where 'RKE' and 'RKR' indicate right kick extend and right kick retract, and 'SL' and 'SR' indicate squat lower and squat raise, etc. As can be seen from Figure 5, the basic algorithm occasionally inserts incorrect segments (for example between samples 7200 and 7300 around the start of the squat motion), and is not as effective at correctly identifying the segment switching point of each action, especially for those actions where few joints are moving (for example, the segment point between squat lower and squat raise around sample 7400). However, correctly identifying these segment switching points is important for observing and later imitating human action, as these switching points are frequently associated with goal states. For example, the switching point of a reaching motion is the location of the object being reached. Adding known motions to the basic segmentation algorithm reduces false positives and improves switching point segmentation for the known motion primitives.

Next, the combined segmentation and clustering algorithm was tested. To evaluate the performance of the algorithm over time, the combined algorithm was tested in epochs, by replaying the motion sequence from start in each epoch. To facilitate the evaluation of the improvements in segmentation as a result of the scaffolding, exemplar motions were added at the start of each epoch, rather than immediately after the motion is abstracted. Figures 6, 7 and 8 show the tree structure following the completion of epochs 1 to 3, respectively. The segmented motions are labeled on the horizontal axis. The first half of the label describes the motion type, where 'LA' stands for Left Arm, 'RA' stands for Right Arm, 'S' stands for 'Squat', 'K' stands for Kick, and 'W' stands for Walk. The second half of the label describes the action being performed during the primitive, for example, during the arm motions, 'L' stands for Lower and 'R' stands for raise. Note that the labels on the horizontal axis were not generated by the algorithm, but were applied manually after analyzing the segments and abstracted motion primitives contained in each node.

After the first epoch, the algorithm has extracted 6 motions, as shown in Figure 6. The abstracted motion abstracted from the group Right Arm Raise ('RAR') is shown in Figure 9. One of the abstracted nodes, the Left Arm Lower ('LAL') group, contains two motions with improper segmentation (i.e., only partial motion is present) as part of the group. In the second epoch, as additional examples of this type of motion are added to the node, a further specialization occurs in the 'LAL' node, adding a child node to the 'LAL' node, as seen in Figure 7. The child node contains a more

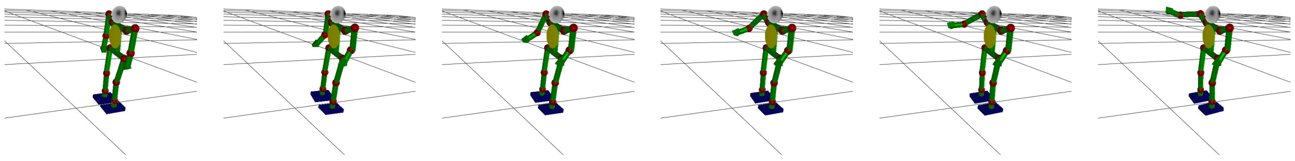

Fig. 9. Frames from the abstracted motion Right Arm Raise, after epoch 1

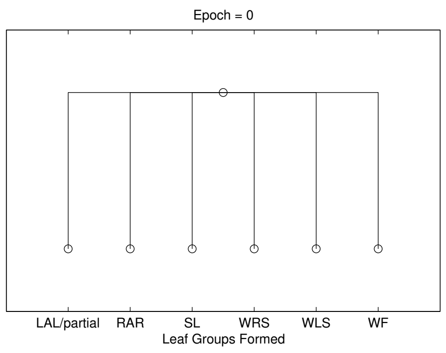

Fig. 6. Motion hierarchy after the first epoch

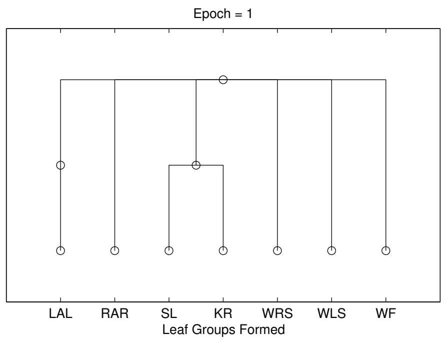

Fig. 7. Motion hierarchy after the second epoch

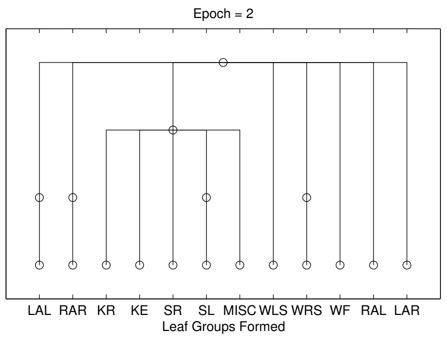

Fig. 8. Motion hierarchy after the third epoch

accurate representation of the 'LAL' movement, containing only correctly segmented examples. Frames of the motion generated from the leaf 'LAL' node from epoch 2 is shown in Figure 10. The algorithm also abstracts the Kick Retract ('KR') motion. The 'KR' motion primitive is placed into the same group as the Squat Lower ('SL'), since these are similar (leg only) motions.

In the third epoch, the algorithm correctly abstracts 4 additional motions, Kick Extend ('KE'), Squat Raise ('SR'), as part of the leg motions parent node, and Right Arm Raise ('RAL') and Left Arm Raise ('LAR') as part of the root node. Frames of the 'KE' motion abstracted from epoch 3 are shown in Figure 11. Note that since the body joint is not used during the clustering and segmentation, the body joint is held in a fixed position when animating the generated motions. In addition, a node is formed containing a series of partial (incorrectly segmented) leg motions, labeled 'MISC' in Figure 8. However, due to the fact that segmenting errors tend to occur quasi randomly, the 'MISC' node maximum distance between exemplars is quite large compared to nodes comparing correctly segmented data, so that this node can easily be excluded from being passed back to the segmentation algorithm, as described in Section 4.

As can be seen from these results, the algorithm quickly and robustly extracts motion primitives in the presence of segmentation inaccuracies. After 3 epochs, corresponding to 12 minutes of observation data, all the presented motion primitives are correctly extracted. Extracted motion primitives are also used to improve segmentation, so that the performance of both the segmentation and clustering improves over time.

Compared to the performance of the standalone clustering algorithm [15], [16] where the provided motions are manually segmented, the combined algorithm results in a deeper tree structure, as there is more variability between the motions initially, due to the automated clustering. As the clustering performance improves, a subgroup of more similar motions forms (i.e., the correctly segmented motions), which is then segmented and added as a leaf node, so that most motion branches contain an upper level node, containing all segment types, and the leaf node, containing the accurate model. On the other hand, when using manually segmented motions as input, only the leaf nodes tended to form, since the variability between motions was much smaller. However, at the leaf node level, the motions extracted were comparable, and did not contain classification errors.

VI. Conclusions

This paper develops an approach towards fully autonomous, on-line, long term incremental learning and hierarchical organization of whole body motion primitives. Motion primitives are autonomously segmented by building a Hidden Markov Model over a window of previous observations, and finding the optimum state sequence over the

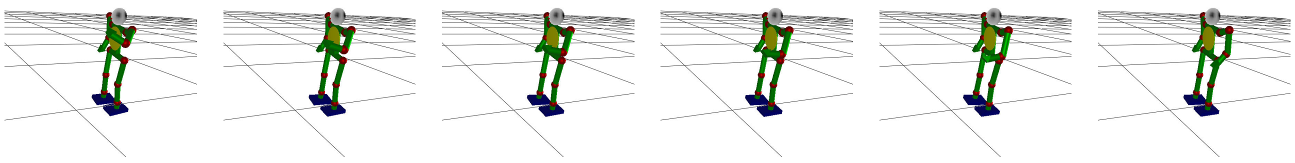

Fig. 10. Frames from the abstracted motion Left Arm Lower, after epoch 1

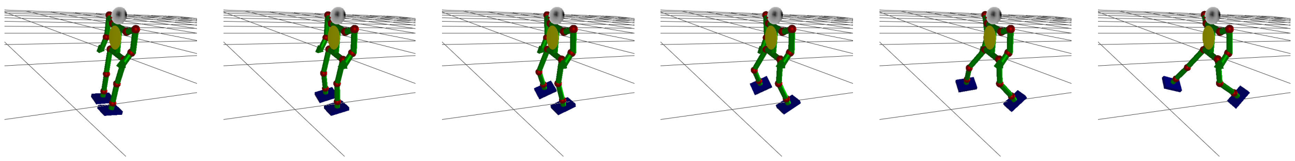

Fig. 11. Frames from the abstracted motion Kick Extend, after epoch 2

model [7], [8]. The optimum state sequence is the desired segmentation result. The segmentation results are further improved by scaffolding the fully unsupervised segmentation method with known motion primitives. Known motion primitives are abstracted from the segmented data by an on-line, incremental segmenting algorithm [15], [16]. Following the observation of each new motion sequence, the observation is placed into the closest motion grouping, based on the model distance between the observation and the group model. The modified group is then analyzed via clustering to extract child nodes, i.e. new, more specific motion primitives. The clustered motions are thereby incrementally 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. The developed algorithm is robust to segmentation errors, and quickly and accurately extracts the presented motion primitives.

VII. 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.

REFERENCES

[1] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[2] 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*, vol. 358, pp. 537 – 547, 2003.

[3] M. Pomplun and M. J. Matarić, "Evaluation metrics and results of human arm movement imitation," in *IEEE-RAS International Conference on Humanoid Robotics*, 2000.

[4] A. Fod, M. J. Matarić, and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous Robots*, vol. 12, no. 1, pp. 39–54, 2002.

[5] J. Lieberman and C. Breazeal, "Improvements on action parsing and action interpolatin for learning through demonstration," in *Proceedings of the IEEE/RAS International Conference on Humanoid Robots*, 2004, pp. 342–365.

[6] N. Koenig and M. J. Matarić, "Behavior-based segmentation of demonstrated tasks," in *Proceedings of the International Conference on Development and Learning*, 2006.

[7] J. Kohlmorgen and S. Lemm, "A dynamic hmm for on-line segmentation of sequential data," in *NIPS 2001: Advances in Neural Information Processing Systems*.

[8] B. Janus and Y. Nakamura, "Unsupervised probabilistic segmentation of motion data for mimesis modeling," in *IEEE International Conference on Advanced Robotics*, 2005, pp. 411–417.

[9] B. Janus, "On-line motion segmentation algorithm for mimesis model," Master's thesis, University of Tokyo, 2006.

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

[11] W. Takano, "Stochastic segmentation, proto-symbol coding and clustering of motion patterns and their application to signifiant communication between man and humanoid robot," Ph.D. dissertation, University of Tokyo, 2006.

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

[13] 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*, 2005, pp. 2900–2905.

[14] ——, "Segmentation, memorization, recognition and abstraction of humanoid motions based on correlations and associative memory," in *IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 1–6.

[15] D. Kulić, W. Takano, and Y. Nakamura, "Incremental on-line hierarchical clustering of whole body motion patterns," in *IEEE International Symposium on Robot and Human Interactive Communication*, 2007, pp. 1016–1021.

[16] ——, "Towards lifelong learning and organization of whole body motion patterns," in *International Symposium of Robotics Research*, 2007, pp. 113–124.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[18] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[19] D. Kulić, W. Takano, and Y. Nakamura, "Representability of human motions by factorial hidden markov models," in *IEEE International Conference on Intelligent Robots and Systems*, 2007, pp. 2388–2393.

[20] K. Kurihara, S. Hoshino, K. Yamane, and Y. Nakamura, "Optical motion capture system with pan-tilt camera tracking and realtime data processing," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1241–1248.