2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

Scaffolding On-line Segmentation of Full Body Human Motion Patterns

Dana Kulić and Yoshihiko Nakamura

Abstract— **This paper develops an approach for on-line segmentation of whole body human motion patterns during human motion observation and learning. A Hidden Markov Model is used to represent the incoming data sequence, where each model state represents the probability density estimate over a window of the data. Based on the assumption that data belonging to the same motion primitive will have the same underlying distribution, the segmentation is implemented by finding the optimum state sequence over the developed model. The basic algorithm is modified to add the capability for modifying the model based on known motion primitives. The inclusion of such scaffolding motion primitives can improve the performance of the basic segmentation algorithm. The modified algorithm is tested on a corpus of continuous human motion data to show the efficacy of the proposed approach.**

I. Introduction

Learning from observation is an attractive proposition for humanoid robots, as the similar body structure to humans can be utilized to bootstrap learning. Many algorithms have been proposed for learning human motions through demonstration and imitation [1], [2]. 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 segmented a priori, and the learning is a one shot, off-line process. In this case, there is no need to autonomously segment or cluster the motion data, as this task is performed off-line by the designer. 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.

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 our previous work [3], [4] we have been developing algorithms for long-term on-line clustering and organization of whole body motion patterns. Our approach is aimed at 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. In this paper, we develop an algorithm for autonomous segmentation of the incoming continuous data stream into whole body motion primitives, which can then be passed to the clustering algorithm. In addition, the proposed segmentation approach can be improved when some of the motion primitives are already known, for example, through manual input by the teacher, or through automated extraction via the clustering algorithm.

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

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 take advantage of known 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, Matarić and colleagues [5], [6] developed several algorithms for segmenting motions based on the velocity properties of the joint angle vector. In Pomplun and Matarić [5], 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. [6], 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. Lieberman and Breazeal [7] improve upon this approach by automating the threshold selection and adding heuristic rules for creating segments when there is contact with the environment. However, with all the velocity based approaches, it becomes more difficult to tune the algorithm as the number of joints increases. For example, it becomes more difficult to select a single threshold for the RMS value of the joint velocities which will accurately differentiate between segments at rest and segments in motions when the dimension space is large and different types of motions (arm motions only vs. motions including arm and leg movement) are considered.

Koenig and Matarić [8] 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 [9] 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. However, the algorithm is not tested on a high DoF data stream, such as full body human motion data. Janus and Nakamura [10], [11] apply this approach to human motion capture data, however no quantitative analysis is applied to the resulting segmentation output.

In the second approach, motion primitives are specified by the designer a-priori, and segmentation is based on the comparison between the known motions and the incoming data. In Takano and Nakamura [12], [13] 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.

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. We would like a method which can begin the learning process in an unsupervised manner, similar to the approach of Janus and Nakamura [10], but which can improve its performance over time, as more motions become clustered and abstracted as motion primitives [3], [4]. The learned motion primitives (or motion primitives which are manually provided to the robot) should be used to improve the segmentation performance.

In the proposed approach, we investigate the use of the Kohlmorgen and Lemm [9] algorithm for unsupervised segmentation of on-line human motion data. While the algorithm achieves acceptable performance in the aggregate, poor performance is observed at those segmentation points where few of the many DoFs are moving. The basic algorithm is modified to improve performance with this type of data. Specifically, as observed motions become known, the basic algorithm is modified to include information about known motion primitives. The known motion primitives are used to modify the HMM model used to generate the optimum state sequence which represents the segmentation result. In particular, known motions are analyzed to determine which joints are most active, and the distance measure is then modified to overweight these joints, improving the segmentation result. Section 2 summarizes the basic Kohlmorgen and Lemm segmentation algorithm [9], [10]. Section 3 describes the modified algorithm. In Section 4, the original and modified algorithm are tested and compared on human motion capture data. Section 5 concludes the paper and provides directions for future work.

II. Unsupervised Probabilistic Segmentation

The Kohlmorgen and Lemm [9] 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,

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

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 L, via a standard density estimator with multivariate Gaussian kernels, centered on the data points in the window $\{\vec{x_{t-i}}\}_{i=0}^{L-1}$.

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

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.

$$\begin{aligned} D(p_{t1}(\mathbf{x}), p_{t2}(\mathbf{x})) = {} & \frac{1}{L^2(4\pi\sigma^2)^{d/2}} \\ & \sum_{i,j=0}^{L-1} [exp(-\frac{(\vec{x_{t1-i}} - \vec{x_{t1-j}})^2}{4\sigma^2}) \\ & - 2exp(-\frac{(\vec{x_{t1-i}} - \vec{x_{t2-j}})^2}{4\sigma^2}) \\ & + exp(-\frac{(\vec{x_{t2-i}} - \vec{x_{t2-j}})^2}{4\sigma^2}) \end{aligned} \tag{3}$$

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:

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

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

The initial state distribution is given by the uniform distribution, and the state transition matrix is designed such that transitions to the same state are k times more likely than transitions to any of the other states.

$$a_{ij} = \begin{cases} \frac{k}{k+N-1} & \text{if } i = j; \\ \frac{1}{k+N-1} & \text{if } i \neq j. \end{cases} \tag{5}$$

where N is the number of states of the HMM. The Viterbi algorithm [14] can then be used to find the optimum state

sequence given the current set of observations. An on-line variant of the Viterbi algorithm is also developed [9], which incrementally builds the state path table as each new state is observed, by re-using the the estimate of the likelihood and optimal state sequence from the previous time step. This assumes that no paths in previous segments include the newly added state. The algorithm is also accelerated by reformulating the Viterbi algorithm so that the cost (i.e., $-\log(L)$) is minimized rather than maximizing the likelihood L. The basic algorithm is outlined in Figure 1. In Figure 1, $C = 2\varsigma^2 \log(k)$ represents the transition cost for for switching to a new 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.

procedure ONLINEVITERBI
  Initialize
  $c_0 \leftarrow o_0 \leftarrow d_{0,0} \leftarrow 0$
  for $t \leftarrow 0, T-1$ **do**
    Find optimum path ending in new state r
    $c_r(t) \leftarrow D_{r,t} +$
    $\begin{cases} 0 & \text{, if } t = 0 \\ \min\left(c_r(t-1), o(t-1) + C\right) & \text{, else} \end{cases}$
    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
    $c_s(T) \leftarrow d_{s,T} + \min\left(c_s(T-1), o(T-1) + C\right)$
  end for
  $o(T) \leftarrow \min_s\left(c_s(T)\right)$
end procedure

Fig. 1. Kohlmorgen-Lemm Algorithm Pseudocode

To prevent the state list from growing to infinity as the number of data points observed increases, Kohlmorgen and Lemm [9] propose removing states following a segment away from that state. However, Janus [11] have found that this approach leads to over-segmenting, as the considered data range becomes too small (on the order of $5L$) 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.

III. SCAFFOLDING THE SEGMENTATION

The basic algorithm functions well when most joints are moving and change the direction of motion at the segment point, but is not as accurate at segmentation points where few joints are moving. The basic algorithm is modified to take advantage of known motions to improve the segmentation result at these segmentation points. Known motions could be provided manually from the user, or through the use of an automated clustering technique [4], [15].

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.

$$D_k(s_t, s_p) = \min_{i=1:n_{sp}} D(p_t, p_{pi}) \tag{6}$$

$D_k(s_t, s_p)$ 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 known motion patterns are then used to 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:

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

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.

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

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_k is the combined variance computed over the entire known motion window.

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 9.

$$a_{ij} = \begin{cases} \frac{k}{C} & \text{if } i = j; \\ \frac{1}{C} & \text{if } i \neq j \text{ and } i \in \mathcal{S}_t; \\ \frac{K_s}{C} & \text{if } i \neq j \text{ and } i \in \mathcal{S}_p. \end{cases} \tag{9}$$

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 the initial implementation of the algorithm, temporary states corresponding to the same motion type as the permanent state were allowed to coexist with the permanent states. However, this caused competition between two very similar states, introducing additional segmentation points and oscillation between the permanent and temporary states. To avoid this problem, temporary states were merged with the permanent state if the distance between them was sufficiently small. This approach is similar to the on-line HMM model learning of Dolan et al. [16].

The modified algorithm is outlined in Figure 2. 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. As before, $c_s(t)$ 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.

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. 2. Scaffolded Segmentation Algorithm Pseudocode

A final modification to the original algorithm concerns the smoothing parameter σ. In the original algorithm (see Equations 2 and 3), σ is calculated for each analysis window, as a means of keeping the distance values roughly similar between cases when a large number of DoFs is moving, and when only few DoFs are moving. However, this online adjustment approach runs into problems when there is little or no motion. In this case, σ can become so small that even differences due to noise or small involuntary movements become segmented, introducing spurious segmentation points. On the other hand, during segments when a large number of DoFs are moving, σ can become so large that all distances are minimized, resulting in no segmentation even when significant motion differences exist. To avoid these problems, a maximum and minimum value of σ are enforced in the modified algorithm.

Because the permanent modified algorithm merges permanent states with similar temporary states, the resulting HMM structure has the same number of states as the basic algorithm. This means that the computation time for the Viterbi algorithm will be unchanged from the basic algorithm. However, the modified algorithm requires additional processing time each new data window is observed, to compare the new observation window to each of the known states, via Equation 6. The total processing time for this additional step will depend on the number of known motions input into the system.

IV. EXPERIMENTS

A large data set was collected in a motion capture studio to test the original and modified algorithms on a lengthy continuous sequence of a variety of whole body motions. Figure 3 shows the motion capture setup and marker locations. A total of 34 markers was used. The data set consists of 20 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 751 motion segments. In some cases, there is a pause between motions, while other motions are fluidly connected. Table I summarizes the motions performed in terms of the extracted segments.

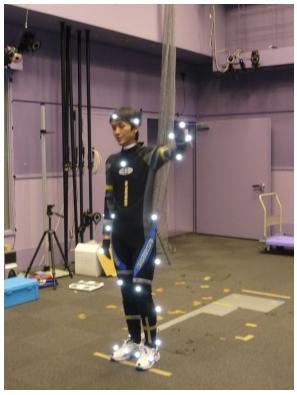

Fig. 3. Motion Capture and Marker Setup

The motion capture system [17] captures the Cartesian position of markers located on the body (for example,

TABLE I

MOTION PRIMITIVES SUMMARY

Motion Description	Motion Label	Number Performed
Right Arm Raise 180 degrees	RAR180	52
Right Arm Lower 180 degrees	RAL180	52
Left Punch Extend	LPE	20
Left Punch Retract	LPR	20
Bow Down	BAD	26
Bow Retract	BAU	26
Left Arm Raise 90 degrees	LAR90	28
Left Arm Lower 90 degrees	LAL90	28
Right Kick Extend	RKE	28
Right Kick Retract	RKR	28
March Left Leg Raise	MLR	26
March Mid Swing	MMID	26
March Right Leg Lower	MRL	26
Left Kick Extend	LKE	23
Left Kick Retract	LKR	23
Both Arms Raise 180 degrees	BAR180	27
Both Arms Lower 180 degrees	BAL180	27
Squat Down	SQD	21
Squat Retract	SQU	21
Right Arm Raise 90 degrees	RAR90	22
Right Arm Lower 90 degrees	RAL90	22
Both Arms Raise 90 degrees	BAR90	26
Both Arms Lower 90 degrees	BAL90	26
Walk Left Leg Raise	WLR	25
Walk Mid Swing	WMID	25
Walk Right Leg Lower	WRL	25
Left Arm Raise 180 degrees	LAR180	3
Left Arem Lower 180 degrees	LAL180	3
Right Punch Extend	RPE	23
Right Punch Retract	RPR	23

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. The motion capture data was also animated in slow motion and segmented manually, for comparison with the automated segmentation results. A sample of the walk motion pattern segments is shown in Figure 4.

The data was first tested on the basic algorithm, described in Section 2. Table II summarizes the parameters used for the algorithm. Table III shows the performance of the basic segmentation algorithm, as compared to the manually segmented results. A segment point was considered correct if it occurred within 4 windows of the manually obtained results. A segment point was counted as a false positive, if it occurred in a section where no manual segment point was specified within 4 windows of the given segment point. A false negative was counted if no segmentation point was specified within a 4 window frame of a manually found segmentation point. As can be seen in Table III, the basic algorithm achieves a correct segmentation rate of about 79%, but also suffers from a high rate of false positives (i.e., additional segmentation points inserted in the middle of a single motion primitive).

TABLE II

ALGORITHM PARAMETERS

Parameter	Value
k	1.25
ς	0.7
m	5
L	5
T	20

TABLE III

SEGMENTATION RESULTS FOR THE BASIC AND MODIFIED ALGORITHMS

Algorithm	Correct	False Pos	False Neg
Basic	594	240	158
Scaffolded (with Bow)	602	241	150
Scaffolded (with Bow and LKick)	610	240	142

Some of these false positives occur when distinctive subsegments occur within a segment which are not marked in the manual results. For example, on some occasions, within the RPE segment, two clear subsegments can be observed: raising both hands into a fighting stance, and then extending the right hand. At other times, the RPE segment is executed as a single smooth movement, with no discernible subsegments.

A section of the segmented data stream is shown in Figure 5. In the figure, the dotted lines represent the manual segmentation points, while the solid lines indicate the segmentation result generated by the basic algorithm. The y axis indicates the action being performed, using segment acronyms as defined in Table I. Figure 6 shows the segmentation results by segment category. As can be seen from Figure 5, the basic algorithm tends to correctly identify the start and end of a complete action, is not as effective at correctly identifying the segment switching point of each action, especially for those actions where few joints are moving. 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.

The basic algorithm is not as effective in segmenting at these points, because only a small subset of the complete set of DoFs is moving. For example, for the bow motion, during most of this motion, only the two hip joints exhibit significant movement. This scenario is much more difficult than the boundary between two different motions, which usually marks the end of the movement of one set of DoFs, and the start of movement of a different set of DoFs.

The data was next tested with the modified segmentation algorithm, initially by adding a set of motion primitives as known examples. For the scaffolded algorithm, the state transition bias towards permanent states K_s were set to 1.05. The merging distance threshold D_t was set to 15% of the maximum distance over the current data. The remaining parameters were set to the same values as the basic algorithm, as detailed in Table II. The known motion primitives

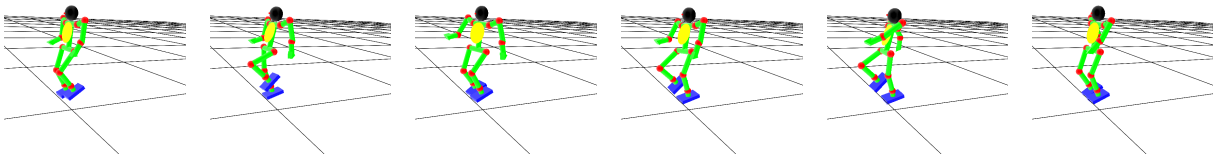

Fig. 4. Sample Walking Motion - Animated with a 20 degree of freedom humanoid model

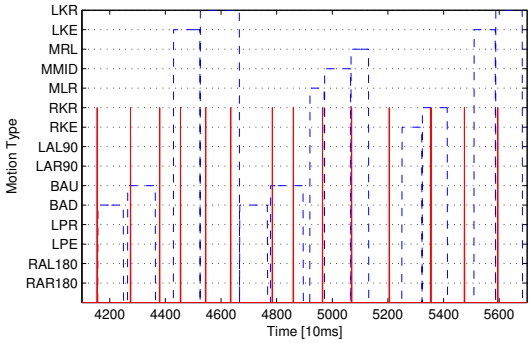

Fig. 5. Segmentation Result Detail for the Basic Algorithm

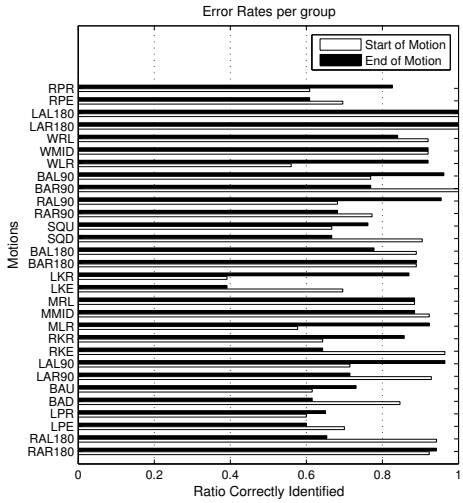

Fig. 6. Segmentation Result for the Basic Algorithm by Motion Segment

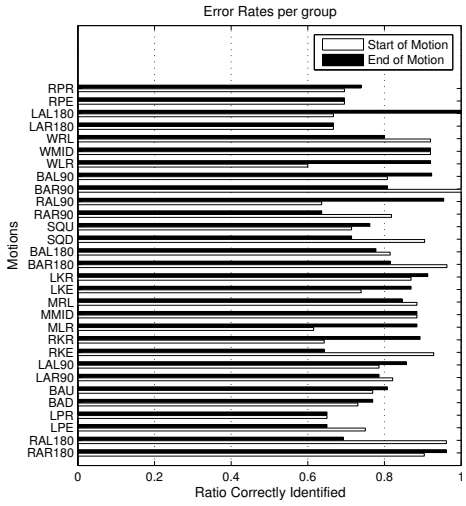

Fig. 7. Segmentation Result for the Modified Algorithm by Motion Segment

were extracted from the motion data set, and manually segmented and provided to the segmentation algorithm. Table III shows the performance of the scaffolded segmentation as compared to the basic algorithm, with the bow motion primitive provided, and with the bow and left kick motion primitive provided. The same evaluation criteria was used to evaluate the scaffolded segmentation results, as for the basic algorithm described above. Still frames of the animated exemplar motion patterns are shown in Figures 8 and 9.

Figure 7 shows the segmentation results by segment category. As can be seen from these results, the modified algorithm improves the segmentation, especially at the critical action switching points. The accuracy at the switching point for the bow motion (the scaffolded motion) is improved from 61% to 77%, and for the kick motion from 39% to 87%.

V. Conclusions and Future Work

In this paper, an approach for automated segmentation of continuous human full body motion data was presented. The basic algorithm [9] is based on the assumption that temporal data belonging to the same motion primitive will belong to the same underlying distribution. Segmentation is achieved by building a Hidden Markov Model over a window of previous observations, and finding the optimum state sequence over the model. The optimum state sequence is the desired segmentation result. In this paper, the basic approach is improved by scaffolding the fully unsupervised segmentation method with known motion primitives. The addition of known motion primitives, which can be obtained either through manual input or automatically through automated clustering [3], [4], improves the segmentation results by improving the accuracy of the segmentation points and reducing the occurrence of false positives.

The segmentation algorithm can be combined with an automated clustering algorithm [15] to enable fully automated human full-body motion segmentation, clustering and hierarchy formation. The automated clustering algorithm can then be used as the source of scaffolding motion primitives for

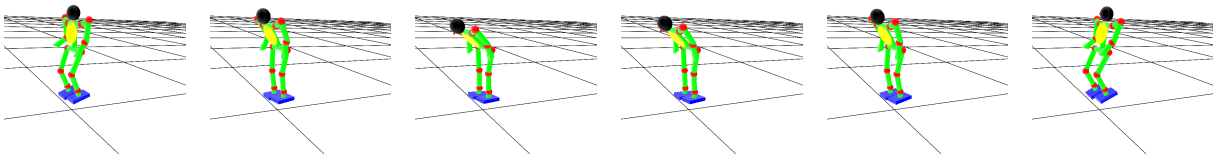

Fig. 8. Sample Bow Motion - Used as a scaffolding exemplar

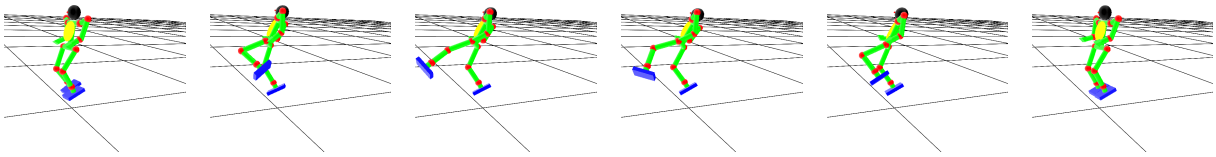

Fig. 9. Sample Left Kick Motion - Used as a scaffolding exemplar

the segmentation algorithm, such that the combined system autonomously improves its segmentation performance over time, as more motions are observed and abstracted.

VI. 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 assistance of Dr. Dongheui Lee, Dr. Wataru Takano and Mr. Akihiko Murai with the data set collection is gratefully acknowledged.

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] 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.

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

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

[6] 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.

[7] 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.

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

[9] J. Kohlmorgen and S. Lemm, "A dynamic hmm for on-line segmentation of sequential data," in *NIPS 2001: Advances in Neural Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14, 2002, pp. 793–800.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] D. Kulić, W. Takano, and Y. Nakamura, "Combining automated on-line segmentation and incremental clustering for whole body motions," in *IEEE International Conference on Robotics and Automation*, 2008, 2591–2598.

[16] K. R. Dixon, J. M. Dolan, and P. K. Khosla, "Predictive robot programming: Theoretical and experimental analysis," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 955–973, 2004.

[17] 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.