2013 IEEE RO-MAN: The 22nd IEEE International Symposium on
Robot and Human Interactive Communication
Gyeongju, Korea, August 26-29, 2013

Motion Learning From Observation Using Affinity Propagation Clustering*

Guoting (Jane) Chang and Dana Kulić

Abstract— **During robot imitation learning, a key problem when observing the motions of a demonstrator is the modeling and recognition of movement prototypes. This paper proposes using Affinity Propagation (AP) to cluster motions modeled using either Dynamic Movement Primitives (DMPs) or Hidden Markov Models (HMMs). The proposed AP clustering algorithm is simple and efficient, provides robust results and automatically identifies representative exemplars for each motion group, leading to a minimal representation of the observations that can also be used to generate motions. In experiments using videos and motion capture data of human demonstrations, it is shown that the weight parameters of the DMP model can be used as features for motion recognition and the proposed method can distinguish between different (coarse distinction) or similar (fine distinction) motion groups.**

I. INTRODUCTION

In order for robots to become useful in human environments, they will need the ability to learn by observing human teachers. This requires the ability to learn motions and motion prototypes, which involves recognizing whether a motion belongs to the same group as a previously learned motion and finding a minimal representation for each group of motions.

This paper focuses on automatically abstracting and representing motions via motion primitives [1] without the need for manual labeling. This is achieved by extracting motion trajectories from continuous time series observation data, segmenting the motion trajectories, modeling the motion segments and clustering them into groups. The motion models are an abstract representation of the observed motions, as they can be used to generate motion trajectories of similar shape to different goal positions. Motions are further abstracted through clustering via Affinity Propagation (AP), a message passing algorithm that identifies the most representative exemplar to serve as each cluster center. The clustering algorithm is simple and efficient, and can robustly identify meaningful motion clusters that match the expected true classes.

A. Related Work

Motion primitive learning is an active area of research, and a number of modeling techniques have been applied for representing and learning motion segments, including neural networks [2], stochastic models [3], [4] and dynamical systems [1], [5]. A common approach is to use stochastic models such as Hidden Markov Models (HMMs) [3] or Gaussian Mixture Models (GMMs) [4]. Variations of HMMs such as Parametric Hidden Markov Models (PHMMs) [6] and Factorial Hidden Markov Models (FHMMs) [7] have also been proposed. The advantage of stochastic models such as HMMs or GMMs is that the models are generative, i.e. can be used to generate motions similar to those that have been learned. However, due to the lack of a mechanism to parameterize the initial and final positions, standard HMMs are unable to generate motions where the shape of the motion trajectory is the same but the initial and final positions vary, and extensions such as PHMMs become necessary at an additional computational cost.

Another approach for modeling motions based on nonlinear dynamical systems is Dynamic Movement Primitives (DMPs) [1], [5]. DMPs can generate generalized motions as the initial and final positions are explicitly parameterized in the DMP formulation [8], [9]. The correlations between weight parameters of the DMP have been shown to reflect similarities between motions [1], [10]. It has also been shown that simultaneous motion recognition and segmentation can be performed using DMPs [11]. However, the approach in [11] requires a pre-existing library of movement primitives to be created.

Similar to [11], many algorithms for learning motions assume that labeling of motion exemplars is provided a priori [12]. Fewer algorithms have used clustering for automatically recognizing and grouping motions. Of these, [3] models motion primitives with HMMs and uses the Kullback-Leibler (KL) distance and Multi-Dimensional Scaling (MDS) to reduce the dimensionality of the data. On the other hand, [7] models motions as FHMMs and uses hierarchical agglomerative clustering (HAC) for grouping and storing motions. However, the resulting dendrogram tree structures are presentation order dependent [7] and online tree correction may be necessary [13] at an additional computational expense. Furthermore, additional computational expense is incurred in training FHMMs to represent a group of motions for motion generation and there is no guarantee that the resulting motion will remain kinematically consistent.

In this paper, we use an exemplar-based clustering method called Affinity Propagation (AP) [14] that leads to stable results with simple iterative computations that is not affected by the order in which motions are presented. Prior knowledge of the number of clusters is not necessary, but if present, can be used to fine-tune the results. We consider motion representation using both DMPs and HMMs. The exemplar based nature of AP is desirable, as the exemplar for each

*This work was supported by the Natural Science and Engineering Research Council of Canada.

G. J. Chang and D. Kulić are with the Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada {g5chang, dkulic}@uwaterloo.ca

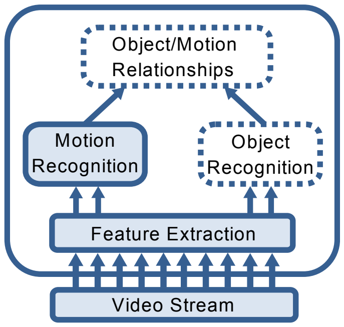

Fig. 1. Framework of the overall learning system. This paper focuses on motion learning for the motion recognition component of the learning system.

cluster is guaranteed to be kinematically achievable.

II. Proposed Approach

The long term goal of this research is to develop a robot learning system shown in Figure 1. The main focus of this paper is on motion learning for the motion recognition component. Motion learning consists of motion segmentation, motion modeling and motion clustering.

A. Motion Segmentation

A simple segmentation technique is used that places a segment point whenever the displacement in one of the observed degrees of freedom switches from non-moving (zero displacement) to moving (non-zero displacement) or vice versa. This motion segmentation strategy is capable of breaking a motion trajectory into meaningful segments when one type of motion switches to another with a transition between stopping and moving or moving and stopping. Only the segments of the motion trajectory that are of interest, i.e. the segments when the demonstrator is actually moving, are used for the subsequent motion modeling and clustering. This simple segmentation technique is appropriate for the low degree of freedom (DoF) motions considered here. For more complex motions, a more sophisticated motion segmentation strategy such as [15] may be adapted.

B. Motion Modeling

Two motion models are considered, Dynamic Movement Primitives (DMPs) and Hidden Markov Models (HMMs). DMPs [1], [5] model motions using non-linear dynamics:

$$\tau\dot{z} = \alpha_z(\beta_z(g - y) - z), \quad z = \tau\dot{y} - f \tag{1}$$

where y is a scalar representing the current position value of a one dimensional trajectory, z is the velocity which includes a non-linear component f plus the rate of change of y, g is the goal position, α_z and β_z are time constants and τ is a temporal scaling factor equal to the time duration of the motion segment.

If $f = 0$, the system in (1) reduces to a linear dynamical system where g acts as a unique point attractor and the system is guaranteed to be stable [5]. To model complex trajectories with nonlinear dynamics, a normalized weighted sum of Gaussians is used to represent f:

$$f = \frac{v\sum_{i=1}^{N} w_i\Psi_i}{\sum_{i=1}^{N}\Psi_i}, \quad \Psi_i = exp\left(-h_i\left(\frac{x}{g} - c_i\right)^2\right) \tag{2}$$

where v is a gating term, w_i are weights to be trained, N is the total number of basis functions used, Ψ_i is a Gaussian with center c_i and spread h_i. In order to reduce the effects of f as the position y approaches the goal g, the gating term v must tend towards zero as y approaches g. This is accomplished by modeling v as a linear dynamical system:

$$\tau\dot{v} = \alpha_v(\beta_v(g_v - x) - v), \quad v = \tau\dot{x} \tag{3}$$

where x is a phase variable, g_v is a point attractor set to zero so that $x = 0$ and $v = 0$ as y approaches g. This is accomplished by setting the time constants α_v and β_v and the temporal scaling factor τ to the same values as α_z, β_z and τ from (1).

For each motion segment, a separate DMP is trained for each DoF of the motion. The weights w_i of each separate DMP are stacked together in one column vector that represents the feature vector for clustering. Feature vectors of different motions are stacked into a matrix used for AP clustering. Once the weights w_i are obtained, a starting position y_0 and a goal position g can be used to compute the corresponding y values of the trajectory using (1). The DMP code used was adapted from existing MATLAB code [1].

The second model, an HMM [16], abstracts the modeled data as stochastic dynamic processes. The dynamics of the motion primitive are modeled by a hidden discrete state variable, which varies according to a stochastic state transition model $A[N_h, N_h]$, where N_h is the number of states in the model. Each state is associated with an output model $B[N_h, K]$, where K is the number of outputs. For continuous motion data, a Gaussian output observation model is used. Given a set of exemplar motions, the Baum-Welch algorithm is used to find the model parameters A, B which produce the exemplar motions with the highest likelihood. A pair of trained models can be compared by using the Kullback-Leibler (KL) distance - a probabilistic distance measure [16]:

$$D(\lambda_1, \lambda_2) = \frac{1}{T}[logP(O^{(2)}|\lambda_1) - logP(O^{(2)}|\lambda_2)] \tag{4}$$

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 distances is used to form a symmetric measure.

C. Motion Clustering

The last step of motion recognition is to cluster the motions using Affinity Propagation (AP). AP is a clustering algorithm that simultaneously considers all data points as possible candidates for being an exemplar for a cluster [14]. Real-valued messages are passed between data points to indicate the suitability of each data point to serve as a cluster exemplar. As these messages are updated iteratively, good cluster exemplars emerge along with data points belonging to each cluster.

The advantage of AP is that it does not require manually labeled data or the number of expected clusters for training. The exemplar clustering nature of AP is very well suited for directly finding the trained motion model that best represents a group of similar motions. Instead of having to store all of the motion models in a motion group, only the exemplar motion model needs to be stored to generate kinematically achievable motions for the entire motion group.

The input to AP is a matrix of similarities between data points, where the element $s(j,k)$ of this matrix indicates how well the candidate exemplar data point k represents the data point j, i.e. the similarity of data point k to data point j. Any distance metric could be used to calculate this similarity, though the negative Euclidean distance is recommended [14]. For the diagonal elements of the matrix of similarities $s(k,k)$, rather than calculating the distance of data point k to itself, a "preference" value is set. The "preference" value incorporates any prior knowledge of the suitability of a particular data point k being a cluster exemplar, where a higher "preference" value indicates a greater suitability. If no prior knowledge is available, all "preference" values are set to the same value such as the median of the input similarities (for a moderate number of resulting clusters) or the minimum of the input similarities (for a small number of resulting clusters) [14].

Two kinds of messages are passed between the data points: "responsibilities" and "availabilities". The responsibility $r(j,k)$ is sent from data point j to candidate exemplar data point k to indicate how confident data point j is in data point k being its exemplar. The availability $a(j,k)$ is sent from candidate exemplar data point k to data point j to indicate how confident k is in being a good exemplar for j. The steps of the AP algorithm are as follows:

1) Initialize all availabilities to zero.
2) Compute and update the responsibility values:

$$r(j,k) \leftarrow s(j,k) - \max_{k':k' \neq k} \{a(j,k') + s(j,k')\} \quad (5)$$

3) Compute and update the availability values:

$$a(j,k) \leftarrow \min\{0, r(k,k) + \sum_{j':j' \notin \{j,k\}} \max(0, r(j',k))\}$$
$$a(k,k) \leftarrow \sum_{j':j' \neq k} \max(0, r(j',k)) \quad (6)$$

4) Repeat 2) and 3) until the exemplar decisions no longer change for a number of iterations, e.g. for 10 iterations. Exemplar decisions are made for each data point j, by finding the value of k that maximizes $a(j,k)+r(j,k)$. If $j = k$, data point j is an exemplar, otherwise k will indicate the exemplar for data point j.

Existing MATLAB code [14] which runs AP offline was adapted. For DMPs, the negative Euclidean distance between columns of the DMP weight matrix is used as the similarity matrix, while the KL distance is used for HMMs.

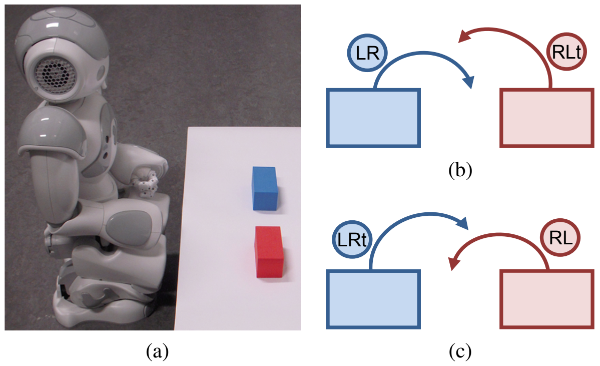

Fig. 2. The experimental setup (a). Two blocks are being stacked by moving (b) the left block first or (c) the right block first. The resulting types of motions are LR (moving a block from left to right), LRt (moving a block from left to right on top of another block), RL (moving a block from right to left), and RLt (moving a block from right to left on top of another block)

III. EXPERIMENTS

The motion learning approach was first tested on a smaller data set of captured video stream data with motions modeled in Cartesian space using DMPs. Next, a larger motion capture data set was also tested using the proposed approach, with motions modeled in joint space using both DMPs and HMMs. A performance evaluation of the AP algorithm compared to commonly used clustering algorithms is also carried out for both data sets.

A. Video Stream Data

Video stream data was captured using the camera of the NAO humanoid robot, which remains stationary during video capture, as shown in Figure 2a. The video always depicts two blocks being moved to the middle of the video frame and stacked. As the video is played, the locations of the blocks are extracted using basic computer vision algorithms. The x and y coordinates of the center of mass of each object (in pixels) are extracted, scaled from pixels to meters and transformed from the camera coordinate frame (x,y) into the frame (z,y) used for motion generation for the NAO.

The goal is for the motion recognition process to extract the four groups of motions shown in Figures 2b and 2c from the captured videos. There are a total of 24 motion exemplars: six instances of each of the four groups of motion were used. Each motion exemplar has a dimensionality of 20, consisting of 10 DMP weights for the y-direction motion segment and 10 DMP weights for the z-direction motion segment. The negative Euclidean distances between the sets of 20 DMP weights per motion exemplar were used as inputs to the AP clustering algorithm.

First, the AP clustering algorithm was run with different preference input values to determine the resulting number of clusters. Using -12.95, the recommended median of all similarities, as the preference value, only two clusters are obtained. These two clusters cleanly divide the motions into those that move from left to right (LR and LRt) and those that move from right to left (RL and RLt). The result of two clusters can be obtained for a wide range of preference values (-170 to -4.2 sampled at step size 0.1).

In order to obtain the more fine-grained division of four motion groups, the preference value must be increased to the range of -3.2 to -0.8 sampled at step size 0.1. Any preference value in this range will work for obtaining the four desired motion groups, which indicates the AP clustering results are stable for this range of preference values. Additionally, all motions continue to be clustered correctly if AP is first run on a smaller set of only four randomly selected instances of each of the four groups of motion and then again on the full data set. This illustrates that AP is able to robustly cluster motions even when new motions are added to the data set.

The confusion matrix of the resulting clustering of the full data set is shown in Table I, where the column headings indicate the true class labels and the first set of rows indicate the clusters found by AP. All values are probabilities. As can be seen from Table I, the LR, LRt, RL and RLt motions were all clustered correctly by AP.

TABLE I
CLUSTERING CONFUSION MATRIX.

Algorithm	**Clusters**	**True Class Labels**			
		LR	**LRt**	**RL**	**RLt**
AP (p = -2)	**LR**	**1**	0	0	0
	LRt	0	**1**	0	0
	RL	0	0	**1**	0
	RLt	0	0	0	**1**
HAC (single)	**LR**	**1**	**1**	0	0
	RL	0	0	**1**	0
	RLt1	0	0	0	**0.833**
	RLt2	0	0	0	**0.167**
HAC (complete, average, minimum-variance)	**LR**	**1**	0	0	0
	LRt	0	**1**	0	0
	RL	0	0	**1**	0
	RLt	0	0	0	**1**
K-means (random, uniform)	**LR**	**1**	**0.167**	0	0
	LRt	0	**0.833**	0	0
	RL	0	0	**1**	0
	RLt	0	0	0	**1**
K-medoids	**LR**	**1**	0	0	0
	LRt	0	**1**	0	0
	RL	0	0	**1**	0
	RLt	0	0	0	**1**

Additional clustering results using hierarchical agglomerative clustering (HAC), K-means clustering and K-medoids clustering are provided in Table I. The HAC algorithm places each data point into its own cluster and then recursively merges them, resulting in a hierarchy of clusters [17] from which the desired number of clusters must be selected. The single-link and complete-link variations of HAC are most commonly used [17], however, the more computationally expensive average-link and minimum-variance may provide more robust results. The K-means and related K-medoids algorithms follow a divisive approach and recursively recompute cluster centers and memberships, where K-means uses the mathematical mean and K-medoids a data point most representative of the cluster as the center. Both the K-means and K-medoids algorithms require the number of clusters to be known beforehand and, as they are not guaranteed to converge to a global optimum, multiple runs must be made to obtain good results. In order to obtain comparative results with the AP algorithm, the Euclidean distance was used as the distance metric for all the comparison clustering algorithms.

All the comparison clustering algorithms, like AP, had no trouble dividing motions into those that move from left to right (LR and LRt) and those that move from right to left (RL and RLt). However, the HAC single-link approach was unable to divide the left to right motions into the respective LR and LRt clusters (i.e. at no point in the cluster hierarchy were LR and LRt cleanly placed into two clusters), instead the RLt motion was divided into two clusters, RLt1 and RLt2, as shown in Table I. The RLt2 cluster only contains one outlier motion exemplar with a slightly different y-axis trajectory that started slower and sped up faster than those of the other RLt exemplars. This difference was enough to cause the minimum distance (the criteria used by single-link HAC) between the outlier RLt and the other RLt exemplars to be larger than that of the LR and LRt exemplars, thus, leading to the incorrect clustering result. As none of the other clustering algorithms use the minimum distance, they were not affected by this outlier.

Similarly, K-means was unable to cleanly divide the left to right motions, resulting in one LRt exemplar being placed into the LR cluster. This mistake persisted even when the best result from 1000 runs of K-means was picked. The misclassified LRt exemplar had a slightly different z-axis trajectory, where the highest point in the trajectory was more pronounced than in the other LRt z-axis trajectories. This difference was enough to cause the outlier LRt to be closer to the mean found for the LR motion cluster than the mean of the LRt motion cluster for the K-means algorithm. The remaining HAC algorithms and K-medoids were able to cluster all the motions correctly.

The representative exemplars selected by K-medoids for both the two and four cluster case are the same as those selected by AP. The motion generation capabilities of the representative exemplars for the four different clusters found are shown in the attached video.

B. Motion Capture Data

The motion recognition system was further validated on a larger, existing motion capture data set provided in [7]. This data set consists of 9 motion groups performed by a single demonstrator (a total of 137 exemplars): walking (WA, 28 exemplars), cheering (CH, 15 exemplars), dancing (DA, 7 exemplars), kicking (KI, 19 exemplars), punching (PU, 14 exemplars), sumo leg raising (SU, 13 exemplars), squatting (SQ, 13 exemplars), throwing (TH, 13 exemplars) and bowing (BO, 15 exemplars). The motion exemplars consist of time series joint angle data that are segmented at the time of recording, i.e. each motion exemplar was recorded separately. Thus, segmentation is not required for this data set. Each motion exemplar in the motion capture data set consists of the joint angle data of a 20 DoF humanoid

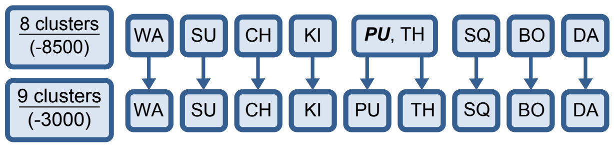

Fig. 3. Clustering results of the motion capture data using DMPs of 5 weights and two preference values for AP. The total number of clusters found is indicated at the beginning of each row (including the preference value used in parentheses). The types of motions contained in each cluster are indicated through their respective labels. If one cluster contains more than one type of motion, the type of motion of the representative exemplar for the cluster is italicized.

model captured at 33 frames per second. A separate DMP with 5 weights was used to model each of the 20 DoF for each motion exemplar, resulting in a dimensionality of 100 for each motion exemplar. The negative Euclidean distances between DMP weight vectors were used as inputs to the AP clustering algorithm.

First, the AP clustering algorithm was run with different preference input values to determine the resulting number of clusters. When the preference value is set to the recommended median of all similarities, exactly 9 clusters are obtained. When the minimum of all similarities is used as the preference value, 8 clusters are obtained. A range of preference values around the recommended median and minimum will lead to the 9 and 8 cluster cases, respectively, indicating that the AP clustering results are stable. AP again clustered all motions correctly when run with only about 70% of the motion exemplars randomly selected from each motion group followed by the full data set.

The results of the AP algorithm clustering the full data set are shown in Figure 3. At 8 clusters, the punching (PU) and throwing (TH) motions were placed into the same cluster with a punching (PU) motion as the representative exemplar. At 9 clusters, punching (PU) and throwing (TH) were correctly placed into their own respective clusters, thereby resulting in the 9 motion groups expected from the data set. All the data points were correctly clustered, i.e. assigned to clusters matching their true classes. The grouping of throwing (TH) and punching (PU) is reasonable due to the similarity between those motions.

Next, a separate 5 state HMM model was trained on each motion exemplar of the motion capture data set. The KL distance between each trained HMM motion model is used as the distance metric for the similarity matrix of the AP algorithm. Similarly to the DMP case, by using the recommended median of all similarities as the preference value, exactly 9 clusters are obtained. When the minimum of all similarities is used as the preference value, 8 clusters are obtained. Again, a range of preference values around the recommended median and minimum will lead to the 9 and 8 cluster cases, respectively, indicating that the AP clustering results are stable. At 8 clusters, punching (PU) and throwing (TH) are placed into the same cluster, with a punching (PU) motion as the representative exemplar. At 9 clusters all motions are clustered correctly into their respective clusters. The DMP and HMM model clustering results are very similar, however, except for the bowing (BO) and dancing (DA) case, AP selected different representative exemplars for the HMM and DMP models. This is likely due to the differences between the distance metrics used.

The comparison clustering algorithms were also applied to the larger dataset. None of the variations of HAC were able to find the 9 desired clusters using the DMP weights as the features. The minimum-variance method fared best, as the 8 cluster case of the HAC minimum-variance method is very similar to the AP results. However, at 9 clusters, the bowing (BO) motion was split into two clusters, while the punching (PU) and throwing (TH) motions were still in one cluster and only at 10 clusters are punching (PU) and throwing (TH) placed into separate clusters. Thus, HAC minimum-variance never fully attains the 9 expected clusters for the large data set. The HAC single-, complete- and average-link performed even more poorly, with bowing (BO), kicking (KI) and cheering (CH) splitting before punching (PU) and throwing (TH) are separated.

Similar behavior is observed when clustering HMM KL distances using HAC single-, complete- and average-link. At 8 clusters, all HAC variations group punching (PU) and throwing (TH) into the same cluster. At 9 clusters, the kicking (KI) motion is separated into two clusters, while punching (PU) and throwing (TH) are still in one cluster. HAC minimum-variance cannot be used with HMM models, as it requires the use of the Euclidean distance.

K-means and K-medoids, on the other hand, were both successful in finding the same 8 and 9 clusters as AP using the DMP weights. However, K-means required a minimum of 50 runs and K-medoids required a minimum of 100 runs in order to consistently determine those clusters. K-means performed better for the motion capture data than the video stream data as the motion capture data contains a much higher number of DoF, which leads to larger Euclidean distances between motion exemplars. K-medoids again found the exact same representative cluster exemplars as AP. For the HMM model, K-means and K-medoids cannot be used, as they need to compute the distances between features and cannot directly take the KL distances as inputs.

The motion generation capability of the representative exemplars was verified using a MATLAB visualization tool of a full-body humanoid model adapted from [18]. Three examples of both the captured data trajectories and DMP generated trajectories are shown in the attached video.

IV. DISCUSSION

The clustering results for both the video stream data and motion capture data indicate that the weights of the DMP can be used as features for motion recognition with just a simple distance metric, e.g. the negative Euclidean distance. The DMP weights can be used to successfully distinguish between different types of motion as well as indicate the amount of similarity between different types of motion, e.g. the punching (PU) and throwing (TH) motions of the large data set were recognized as being more similar to each other than other motions.

TABLE II

COMPUTATION COMPLEXITIES. n IS THE NUMBER OF DATA POINTS, m IS THE NUMBER OF ITERATIONS, K IS THE NUMBER OF CLUSTERS AND l IS THE NUMBER OF RUNS.

Algorithm	**Computational Complexity**	**Source**
AP	O(n^2m)	[14]
HAC	O(n^2log$n$)	[17]
K-means	O(nKl)	[17]
K-medoids	O($K(n-K)^2l$)	[19]

The experimental results also show that AP can be used for motion recognition. The algorithm can be used without any prior knowledge of the number of clusters. AP is simple to use since its only parameter, the preference value, can be computed by simply taking the median of the all the input similarity values. Other clustering methods, such as K-means and K-medoids, require the number of classes to be known beforehand, while HAC requires selecting the level of hierarchy matching the desired number of clusters.

If additional knowledge is available, the preference value of the AP algorithm may be tuned to obtain a coarse or fine division of motions. For the video stream data, both the two and four cluster cases found using AP contain no clustering errors, while the HAC single-link variation and K-means fail to identify the finer distinction between motions in the four cluster case. Additionally, AP works for different motion models and distance metrics, i.e. DMP with negative Euclidean distance and HMM with KL distance. The HAC minimum-variance, K-means and K-medoids cannot directly be used with the KL distance metric for HMMs.

The theoretical computational complexities of the clustering algorithms are shown in Table II. HAC has the highest computational complexity. K-means, on the other hand, appears to be the most efficient, but only if the number of runs l needed to obtain good clustering results is low. Similarly, AP and K-medoids look comparable in terms of computational complexity, but again this may not hold if the number of runs l that K-medoids requires is high. The number of iterations m for AP was found to be more or less constant for the small and large data sets as the large data set required only one more iteration than the small data set to obtain all the clusters. Actual runtimes of the clustering algorithms are not reported, because the algorithms were implemented with different levels of optimization.

AP finds representative exemplars for the motion cluster that can directly be used for motion generation. K-medoids is the only other clustering algorithm amongst those tested that will also find representative exemplars. The representative exemplars found by both AP and K-medoids are the same for the DMP motion models for both the video stream and motion capture data.

V. CONCLUSIONS AND FUTURE WORK

This paper presented an approach for unsupervised motion recognition via clustering. Motions were modeled using DMPs and HMMs, and motion clustering was performed using the AP clustering algorithm. For DMP models, the negative Euclidean distance between the DMP weights was used as a distance metric, while for HMM models, the KL distance was used. The results confirm that the DMP weights can be used as features for motion recognition. Further, comparison with other clustering algorithms shows that the proposed clustering approach is simple to use and efficient, making it an attractive choice for potential real-time applications. Lastly, AP finds exemplars for each cluster, thus, directly selecting the representative motion for the entire cluster which can be stored and used to generate motions for the entire cluster. Future work may involve extending the AP algorithm to run incrementally.

ACKNOWLEDGMENT

The authors would like to thank Professor Ali Ghodsi for suggesting the use of the Affinity Propagation algorithm.

REFERENCES

[1] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Learning attractor landscapes for learning motor primitives,” in *NIPS*. MIT Press, 2003, pp. 1523–1530.

[2] T. Ogata, S. Sugano, and J. Tani, “Open-end human robot interaction from the dynamical systems perspective: Mutual adaptation and incremental learning,” *Adv. Robotics*, vol. 19, no. 6, pp. 435–444, 2004.

[3] W. Takano and Y. Nakamura, “Humanoid robot’s autonomous acquisition of proto-symbols through motion segmentation,” in *Humanoids*, 2006, pp. 425–431.

[4] S. Calinon, F. Guenter, and A. Billard, “On learning, representing, and generalizing a task in a humanoid robot,” *SMC, Part B*, vol. 37, no. 2, pp. 286–298, 2007.

[5] S. Schaal, “Dynamic movement primitives - a framework for motor control in humans and humanoid robotics,” *AMAM*, pp. 261–280, 2006.

[6] D. Herzog, V. Krueger, and D. Grest, “Parametric hidden markov models for recognition and synthesis of movements,” Aalborg University Copenhagen, Tech. Rep., 2008.

[7] D. Kulić, W. Takano, and Y. Nakamura, “Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains,” *IJRR*, vol. 27, no. 7, pp. 761–784, 2008.

[8] A. Gams and A. Ude, “Generalization of example movements with dynamic systems,” in *Humanoids*, 2009, pp. 28–33.

[9] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, “Movement templates for learning of hitting and batting,” in *ICRA*, 2010, pp. 853–858.

[10] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots,” in *ICRA*, 2002, pp. 1398–1403.

[11] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, “Movement segmentation using a primitive library,” in *IROS*, 2011, pp. 3407–3412.

[12] C. Breazeal and B. Scassellati, “Robots that imitate humans,” *TiCS*, vol. 6, no. 11, pp. 481–487, 2002.

[13] D. Kulić and Y. Nakamura, “Incremental learning and memory consolidation of whole body human motion primitives,” *Adapt. Behav.*, vol. 17, no. 6, pp. 484–507, 2009.

[14] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *Science*, vol. 315, pp. 972–976, 2007.

[15] D. Kulić, D. Lee, and Y. Nakamura, “Whole body motion primitive segmentation from monocular video,” in *ICRA*, 2009, pp. 3166 –3172.

[16] 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.

[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] S. Kajita, *Humanoid Robot*. Ohmsha, 2005, in Japanese.

[19] D. Cao and B. Yang, “An improved k-medoids clustering algorithm,” in *ICCAE*, vol. 3, 2010, pp. 132–135.