Automatic Human Mocap Data Classification

Harshad Kadu and C.-C. Jay Kuo, *Fellow, IEEE*

***Abstract*—Automatic classification of human motion capture (mocap) data has many commercial, biomechanical, and medical applications and is the principal focus of this paper. First, we propose a multi-resolution string representation scheme based on the tree-structured vector quantization (TSVQ) to transform the time-series of human poses into codeword sequences. Then, we take the temporal variations of human poses into account via codeword sequence matching. Furthermore, we develop a family of pose-histogram-based classifiers to examine the spatial distribution of human poses. We analyze the performance of the temporal and spatial classifiers separately. To achieve a higher classification rate, we merge their decisions and soft scores using novel fusion methods. The proposed fusion solutions are tested on a wide variety of sequences from the CMU mocap database using five-fold cross validation, and a correct classification rate of 99.6% is achieved.**

***Index Terms*—Database management, human motion analysis, machine learning, mocap data, motion recognition, n-fold cross validation, suffix array, SVM, vector quantization.**

I. Introduction

THE increasing demand for rendering smooth and plausible 3D motion is fueling the development of motion capture (mocap) systems. This new format of high quality 3D motion data has paved its way into animation movies, high-end computer games [1], biomechanics, robotics, gait analysis and rehabilitation [2] and machine translation of sign languages [3], [4]. The diverse applications of mocap data and the rapid development of mocap systems have resulted in a large corpus of motion capture data in recent years. However, the large amount of raw data files makes it difficult to organize. It is desirable that both file names and the associated metadata should be able to provide a high level description of the contents of mocap sequences. Since manual annotation of mocap sequences is labor intensive, it is essential to develop an automated technique that can segment mocap data into homogeneous intervals, classify each interval into a basic motion type, and index it for future retrieval. With such annotated mocap databases in place, a proper motion editing and authoring tool can be used to synthesize realistic motion sequences.

To develop efficient indexing and retrieval techniques for mocap databases, the first step is to classify the data into subsets according to their similarities, which is known as the mocap data classification problem. As far as the human full-body motion is concerned, the mocap system records the actions of human actors by placing several markers on their body. These markers and a grid of infrared cameras help determine the full-body movement of those actors in terms of joint orientations. This joint orientation information constitutes the mocap data used in our research. Human mocap data are essentially time series of human body poses (Fig. 1). Different human motions may be of different time duration, tempo and style. The same motion type, e.g., walking, can vary from person to person. Moreover, visually similar motions may start or end with different joint orientations or even with different human body postures. A robust classification algorithm has to compensate for these irregularities in the data.

In our tree-structured vector quantization (TSVQ) based scheme, the dynamics of human motion is represented by a sequence of multi-resolution codewords. Apparently, two motions are similar if their corresponding codeword sequences are similar. To leverage this similarity, we analyze the distribution of these codewords in the temporal and spatial domains separately. Then, to overcome the individual limitations of these two approaches, we fuse their outcomes and soft scores to make final decision. We examine the proposed solutions on a wide range of motions from the CMU mocap dataset [5]. These motions are recorded in a controlled environment with only one performer per clip. Furthermore, the clips are assumed to be homogeneous, wherein every motion clip belongs to only one motion category.

It is worthwhile to mention that a mocap data classification technique using hierarchical codewords was described in [6], which is a 2-page summary of some intermediate results of our research. The algorithmic description was brief and experimental results were preliminary. Compared to [6], the current work has a more complete treatment on this topic. Specifically, it has the following major contributions:

- Development of an advanced temporal domain approach to mocap data classification using codeword sequence matching;
- Development of a novel spatial domain approach for human mocap data classification with pose-histogram-based SVM classifiers;
- Performance evaluation on a larger dataset containing 278 human motion clips (∼0.5 million frames) spanning 30 motion categories;

Manuscript received January 15, 2014; revised June 05, 2014; accepted September 22, 2014. Date of publication September 29, 2014; date of current version November 13, 2014. This work was supported in part by the University of Southern California Center for High-Performance Computing and Communications. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gokhan Tur.

The authors are with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: hkadu@usc.edu; cckuo@sipi.usc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2014.2360793

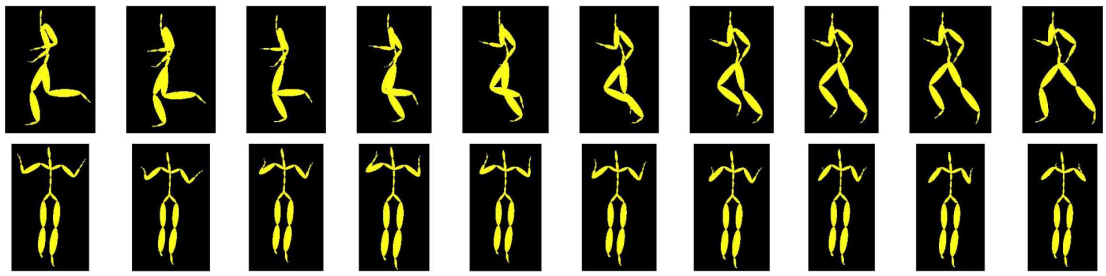


Fig. 1. Illustration of ten consecutive poses of the running motion (top) and the teapot nursery rhyme rendition (bottom).

- Achieving a correct classification rate of 99.6% for a data set comprising of simple, complex as well as perceptually similar motion types.

The spatial domain approach takes the human skeleton structure into account so that it is restricted to human motions only. In contrast, the temporal domain approach does not assume any inherent structure in the data so it is more generic in this sense. Moreover, other formats of marker-based mocap data can be handled with the proposed algorithms if they can provide the 3D pose information. We will describe these new approaches, their implementations, fusion and experimental results in greater detail to offer readers a complete picture of our solution and its superior performance.

The rest of this paper is organized as follows. Related previous work is reviewed in Section II. Our advanced motion classification techniques are described in Section III. Experimental results are presented and analyzed in Section IV. Finally, concluding remarks and possible future extensions are summarized in Section V.

II. Review of Previous Work

Different methods have been proposed to synthesize realistic human motions by reusing existing mocap data via data-driven motion editing and authoring, e.g., Kovar and Gleicher [7], Safonova *et al.* [8], Ren *et al.* [9], Lee *et al.* [10], Zordan *et al.* [11], and Pullen and Bregler [12]. In order to reuse existing mocap data, efficient searching, indexing and browsing techniques are required. So the first step is to group motion clips into subsets based on their similarities, which is the focus of our current research.

Identifying similar motion types in a mocap database has been considered a challenging problem. Most of the previous work on motion comparison adopted features that were close to the raw data. For example, Liu and Popović [13] included the principal component analysis (PCA) reduced raw data, 3D point clouds, joint angles, 3D positions, etc. Forbes and Fiume [14] used the weighted PCA for motion comparison. The performance was further enhanced by integrating the weighted PCA with other features such as characteristic points, seed points, etc. Kovar and Gleicher [7] proposed a numerical similarity technique to find similar motion clips from a large data set. Motion similarity is a semantic concept. The raw numerical data may differ a lot even for two visually similar motions. Hence these types of features do not work well in practice.

Liu *et al.* [15] used automatically extracted key frames and a hierarchical tree of clusters of motions to search similar motions in the database. Sakamoto *et al.* [16] proposed a motion map method that trains the self-organizing-map (SOM) with motion data and indexes clips by SOM nodes. The motion map transforms the N-dimensional data to a 2D map and computes similarity using this map. However, this scheme lacks higher level data indexing and demands exact node-to-node matching in search.

Human motion is typically a long time series with no overt segmentation information so the dynamic time warping (DTW) based motion comparison techniques (e.g., Chiu *et al.* [17], Kovar and Gleicher [7], Wu *et al.* [18]) suffer from the complexity of finding the corresponding start/end points of motion data series for motion similarity comparison. Hsu *et al.* [19] used an iterative motion warping (IMW) technique, which is an improvement over the traditional DTW technique, to find the correspondence by minimizing an objective function with dynamic programming. Generally speaking, these features work well only for certain applications.

Several algorithms were developed to identify locally similar segments in the motion dataset, for example, Arikan and Forsythe [20], Kim *et al.* [21], Kovar and Gleicher [22], Lee *et al.* [10], and Wang and Bodenheimer [23]. Arikan *et al.* [24] proposed a semi-automatic annotation technique using SVM classifiers. Cardle *et al.* [25] used the GEMINI framework to search mocap data sets. Yang and Shahabi [4] proposed a method called EROS to calculate similarity using PCA and eigenvalues. Li *et al.* [26] used the kWAS method to measure similarity of motion clips via singular value decomposition.

Relational features [27]–[30], built upon the Boolean relations between various body parts, are more suitable for generic applications. Common relational features include: the position of joints with respect to a certain plane, the angle between two limbs, the angular velocity of joint angles, the position of joints with respect to other joints, the direction of motion, touch, etc. Such features are more robust to pose distortion and other small aberrations in motion sequences. Although relational features provide a good alternative, automatic selection of a good set of relational features for a wide range of motion types is not straightforward. This imposes a limitation on the utility of these features.

Wu *et al.* [31] presented a cluster-based scheme for mocap data indexing and retrieval, where temporal variations were accounted for while spatial variations were not captured effectively. Later, Wu *et al.* [32] incorporated the spatial complexity of mocap data using hierarchical clustering. But this scheme cannot effectively differentiate between perceptually similar

motions, such as *running* and *marching*. The different variations of the same motion type (e.g., cartwheel) were also not clustered correctly. The SOM representation used in their scheme transforms the 62 dimensional mocap data into a crude 2D representation, which suffers from self occlusions. This significantly hinders its capability to categorize complex motions.

The main contribution of our work is automatic classification of perceptually similar mocap sequences based on the "multi-resolution string representation" that incorporates both spatial and temporal information effectively. It will be shown later in the experimental section that the proposed algorithm offers a correct classification rate of 99.6% for a database consisting of 30 different motion types. To the best our knowledge, no other algorithm with better performance on such a diverse dataset has been reported in the literature.

Although our solution is promising, it has some limitations. For example, it might not work well on a clip containing motions from different categories, e.g., a clip with walking motion followed by running. However, with mocap data segmentation as the pre-processing step, this problem can be mitigated. The proposed solution was not thoroughly tested on categories with minor variations, such as, *get up from the floor without the hands touching the floor* and *get up from the floor by pushing hands against the floor*. As these categories are practically the same, we combined them into a single category in the experiment.

Researchers from the computer vision community have proposed a variety of methods for action recognition in unconstrained video databases such as UCF11 and UCF50. These methods used template matching, finite state models, bag of features, face detection, context, speech and text information etc. for action recognition [33], [34]. But here, we work directly on the human skeleton posture information, which is available with the mocap data.

III. Proposed Classification Methods

We consider four mocap sequence classification methods.

- *Method-A: Motion-String Similarity*: The information about the temporal order of the poses is exploited in motion string similarity comparison using the suffix array technique. But due to the coarse quantization used, Method A is rigid and less robust. To overcome this problem, Method-B incorporated finer details.
- *Method-B: Pose-Histogram Classifier*: TSVQ helps to compute the multi-resolution pose histogram for each motion clip. Multiple binary SVM classifiers, trained using these histograms, are used to generate soft scores for each test motion clip. One weakness of this method is that it does not use the temporal information at all. But Method-A compensates for that.
- *Method-C: Two-Step Score Fusion*: The soft decision scores obtained from methods A and B are combined to make the final decision.
- *Method-D: Two-Step SVM Fusion*: A more advanced SVM based fusion is used to combine the soft scores of methods A and B.

The details are given in the following subsections.

TABLE I
Partitioning of A Human Skeleton for Classification

Methods	Parts	No. of DOFs	Index Range
Method-A	Full-body	59	4-62
	Torso	21	4-24
	Right hand	12	25-36
Method-B	Left hand	12	37-48
	Right leg	7	49-55
	Left leg	7	56-62

A. Method-A: Motion-String Similarity

Human mocap data is a high-dimensional time series where the data at each time instance (called a frame) represents the spatial location of markers in the 3D space [32]. To eliminate the effect of bone sizes, this spatial data is converted to the rotational angles of joints. In simplified terms, the 3D trajectory of placed markers is stored as the trajectory of each degree of freedom (DOF) of the joint, over time [35], [36]. Suppose that $r(t)$ is the 3D position of the root joint at time t, a_i is a scalar or vector representing the orientations of all the DOFs of joint i and N is the total number of joints in the human skeleton $(N = 29)$. Then, a frame $f(t)$ corresponding to time t in the mocap sequence is a 62 dimensional vector which can be mathematically represented as

$$f(t) = \{r(t), a_1(t), a_2(t), \cdots, a_N(t)\}, t = 1, 2, \cdots \quad (1)$$

The first three dimensions (index range 1-3) representing the position of the root joint are neglected, as they are irrelevant to the 3D pose. The remaining 59 elements of vector $f(t)$ (index range 4-62) together form the "full-body" data vectors as shown in Table I. We apply the Tree-Structured Vector Quantization (TSVQ) technique [37], [38] to the full-body data vectors in order to generate a full-body multi-resolution codebook as described below. The TSVQ performs clustering of full-body data vectors by repeated application of the Generalized Lloyd Algorithm (GLA). Iteratively, each cluster is split into two and fine tuned with another round of repeated application of the GLA to the perturbed centroids. After convergence, the parent codeword spawns two child codewords in the tree structure. Finally, a balanced binary tree-like structure is obtained. Fig. 2 shows the full-body level-n codebooks obtained using the TSVQ technique up to level $n = 3$. Owing to balanced binary tree structure, level n in the tree has 2^n codewords. All the full-body codewords at level-n together form the level-n codebook.

This full-body level-n codebook encodes the mocap sequence by replacing each full-body pose vector with a codeword that is closest to it in terms of the Euclidean distance. For example, ten consecutive full-body poses of the running motion are shown in the top row of Fig. 1. If this partial sequence is encoded using a full-body level-9 codebook, each pose will be mapped to a codeword. Due to quantization, the mapped codewords can be the same for a set of consecutive frames. As a result, the mocap sequence can be compactly represented by a series of triplets (namely, a full-body codeword index number and the frame range). This series of codeword indices is called the "motion string". An illustration of the motion string representation is given in Fig. 3, which shows all frames in the aforementioned running motion sequence. The black

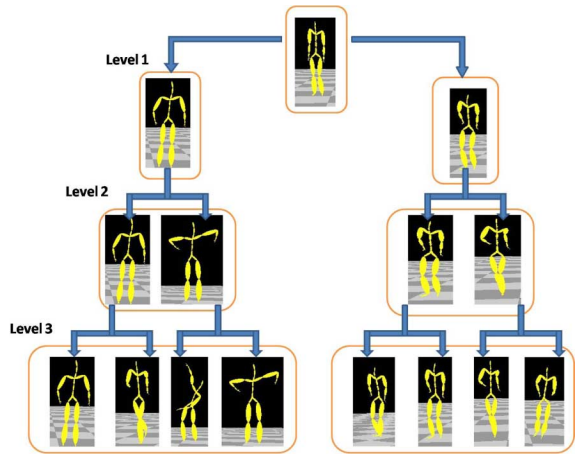


Fig. 2. An illustration of full-body multi-resolution TSVQ codebooks.

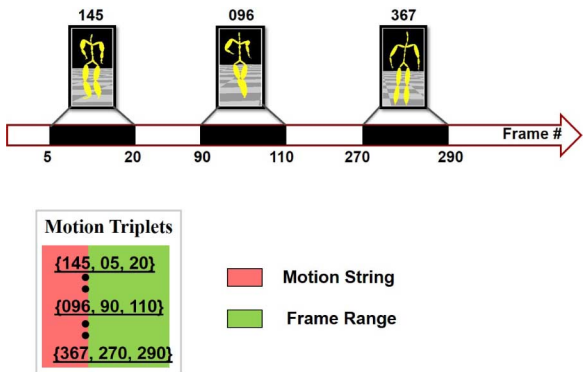


Fig. 3. An illustration of the motion-string representation.

horizontal segments indicate the group of consecutive frames that are mapped to the same full-body level-9 codeword. For the level-9 codebook, the codeword index ranges from 0 to 511 ($= 2^9 - 1$).

Apparently, two motions are similar if their corresponding motion strings are similar. The problem of finding similarity between two motion strings can be reformulated to a string matching problem. The latter can be efficiently solved by using the suffix array technique [39], [40], [41]. In this technique, all suffixes of a string, S, are sorted in an ascending order. The sorted suffixes for a sample motion string, $S = [25; 18; 87; 160; 98; 25; 18]$, are shown in the last column of Table II. Note that the end of the string is denoted by a unique symbol '$'. The '$' symbol is assigned to have the highest value among all codeword indices in the motion string. The suffix notation S_q represents the sub-sequence of S starting from $S[q]$ till the end of the string, followed by $. Only the starting point *i.e.* index 'q' is enough to represent this suffix S_q. The array of indices 'q' defines the "suffix array", denoted by *suftab* in Table II. We have $suftab[\tilde{i}] = q$ for suffix S_q that is at the $\tilde{i}^{th}$ position in the sorted list starting with $\tilde{i} = 0$. To give an example, S_4 represents the sub-sequence starting from $S[4] = 98$, which is $[98, 25, 18, \$]$. The suffix S_4 is at the fifth location ($\tilde{i} = 5$) in the sorted list. Therefore, $suftab[5] = 4$ in Table II.

For a sequence of length p, the suffix array can be built in $\mathcal{O}(p \log p)$ time using an appropriate sorting algorithm. Besides,

TABLE II
THE SUFFIX ARRAY FOR MOTION STRING: [25, 18, 87, 160, 98, 25, 18, $]

$\tilde{i}$	$suftab[\tilde{i}]$	$lcptab[\tilde{i}]$	$bwttab[\tilde{i}]$	$S_{suftab[\tilde{i}]}$ *or* S_q
0	1	0	25	18 87 160 98 25 18 $
1	6	1	25	18 $
2	0	0	-	25 18 87 160 98 25 18 $
3	5	2	98	25 18 $
4	2	0	18	87 160 98 25 18 $
5	4	0	160	98 25 18 $
6	3	0	87	160 98 25 18 $
7	7	0	18	$

Kim *et al.* [42] proposed a linear time complexity algorithm for generating the suffix array. Other supplementary arrays are also constructed to enhance and optimize the string matching functionality. These arrays include:

1) The Burrows Wheeler Transform array, denoted by "bwttab", that stores the element before the first element of the corresponding suffix S_q. The $\tilde{i}^{th}$ element of bwttab is $bwttab[\tilde{i}] = S[suftab[\tilde{i}] - 1] = S[q - 1]$. For example, $bwttab[5] = S[suftab[5] - 1] = S[3] = 160$.
2) The longest common prefix array, denoted by "lcptab", that stores at the $\tilde{i}^{th}$ location, the length of the longest common prefix between $S_{suftab[\tilde{i}]}$ and $S_{suftab[\tilde{i}-1]}$. For example, the longest common prefix of $S_{suftab[3-1]}$ (*i.e.* S_0) and $S_{suftab[3]}$ (*i.e.* S_5) is [25], [18]. Therefore, $lcptab[3] = 2$.

The value stored in $lcptab[\tilde{i}]$ is the length of unique repeated sub-sequence when we have

$$bwttab[\tilde{i} - 1] \neq bwttab[\tilde{i}]. \tag{2}$$

Thus, there is only one unique repeated sub-sequence in this example, which is of length 2 and the sub-sequence is [[25], [18]. In this manner, all unique matches within string S can be found using the suffix array technique.

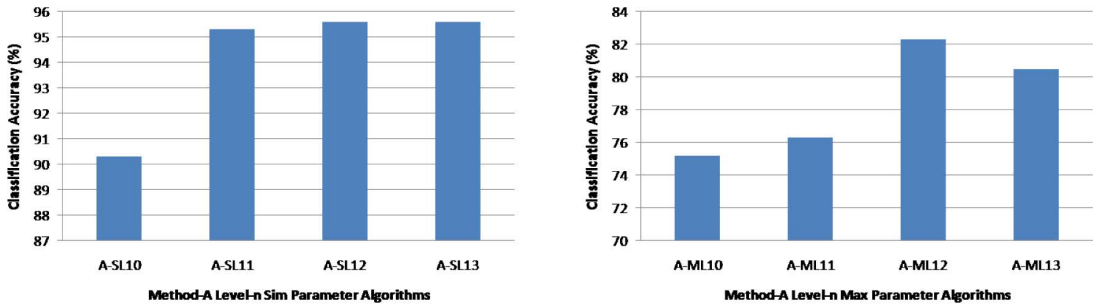
To find similarity between two strings $S1$ and $S2$, they are concatenated into a new string S_c, denoted by $S_c = S1\#S2\$$, where symbol '#' is a unique logical separator between $S1$ and $S2$ but different from '$'. Symbol '#' is assigned the second largest value in string S_c after '$'. The same technique used to find the repeated sub-sequences for a single string is now applied to new string S_c with additional care. That is, one of the matching sub-sequence should begin before the '#' symbol in S_c and the other should begin after the '#'. It ensures that the match is not within $S1$ or $S2$ itself, but across strings $S1$ and $S2$. This additional constraint can be expressed in the following form:

$$\begin{aligned} suftab[\tilde{i} - 1] < |S1| < suftab[\tilde{i}], \ or \\ suftab[\tilde{i}] < |S1| < suftab[\tilde{i} - 1] \end{aligned} \tag{3}$$

where $|\cdot|$ is the cardinality of string $S1$.

In our experiments, all training and testing motion sequences are converted to their motion strings. A test motion string is compared with all training motion strings individually, using the suffix array technique discussed above and the following three parameters are computed for each testing-training string pair:

- *MLM*: the product of ratios of the *M*aximum *L*ength *M*atches to the total length, for the test and the training



(a) (b)

Fig. 4. Standalone classification performance of the motion string similarity comparison: (a) overall performance using the sim parameter; (b) overall performance using the max parameter.

strings. The maximum length sequence 'l' is the maximum value in the lcptab array that satisfies Eqs. (2) and (3).

- *TEM*: the *T*otal number of *E*lements that *M*atch in the two strings. The total number of elements 't' is the sum of all the non-zero values in lcptab array that satisfy Eqs. (2) and (3).
- *SRP*: The *P*roduct of *S*imilarity *R*atios of the test string and the training string.

For example, the ith test string has f codeword indices in its motion string while the jth training string has g codeword indices. The t codeword indices in these two strings are identical and the largest common sequence is of length l. Then $TEM_{i,j} = t$ and

$$MLM_{i,j} = (l/f \times 100)(l/g \times 100)$$
$$SRP_{i,j} = (t/f \times 100)(t/g \times 100). \quad (4)$$

As the calculation of $MLM_{i,j}$ and $SRP_{i,j}$ has its bottleneck in the suffix array formulation, their computational complexity is similar to that of the suffix array, with $p = f + g$. On the average, it takes around 0.5 ms to compare two strings using the suffix array technique on a 64-bit Windows Vista operating system with 4 GB RAM and Intel Core 2 Duo CPU T6500 2.10 GHz.

For the ith test string, the following two metrics are calculated with respect to a given motion category k using the parameters in Eq. (4):

1. *Max*-parameter: the average of MLMs of all training motions in category k;
2. *Sim*-parameter: the average of the similarity product of all training motions in category k.

For the kth category, we have

$$MAX_i^k = \frac{\sum_{j\in k} MLM_{i,j}}{\sum_{j\in k} \mathbb{1}(j \in k)}$$
$$SIM_i^k = \frac{\sum_{j\in k} SRP_{i,j}}{\sum_{j\in k} \mathbb{1}(j \in k)} \quad (5)$$

where $\mathbb{1}(\cdot)$ is the indicator function.

In standalone Method-A, the category with the highest parameter value wins. Alternatively, for each test motion, all categories are pitted against each other, one-on-one, and the category with the higher value for the parameter wins the vote. All category-specific votes are aggregated together and, then, normalized to get the soft scores. That is, we have

$$Vote_{i,MAX}^k = \sum_{\forall m \neq k} \mathbb{1}(MAX_i^k > MAX_i^m)$$
$$Vote_{i,SIM}^k = \sum_{\forall m \neq k} \mathbb{1}(SIM_i^k > SIM_i^m) \quad (6)$$

for the kth category with $m \in \{1, 2 \cdots 30\}$. The standalone Method-A classification results are presented in Fig. 4. The full-body level-n codebooks that offer the best classification results in standalone Method-A are short-listed for fusion to yield the final decision.

B. Method-B: Pose-Histogram Classifier

Dynamic human behaviors are intrinsically low dimensional since legs and hands work in a coordinated fashion [8]. According to biomechanics, a human skeleton can be decomposed into five functionally independent body-parts: right hand, left hand, right leg, left leg and torso. The mocap data inherently provides a clear demarcation between these body-parts. In other words, different portions of the mocap data vectors $f(t)$ can be associated with different body-parts. The first 3 elements give the root joint location. The next 21 elements (i.e. with index number 4-24) provide the information about the angular orientations of joints in the torso, the next 12 elements (i.e. with index number 25-36) for right hand and so on. The number of DOFs for each body-part and their associated index range in the vector $f(t)$, are shown in Table I.

To leverage the spatial symmetry of human bodies, we propose a scheme that partitions a human skeleton into five body-parts using the information from Table I. Then, TSVQ is applied to the DOF vectors of each body-part separately to generate five balanced tree structured codebooks, one for each limb (*i.e.* right hand, left hand, right leg and left leg) and one for the torso. These five body-part codebooks provide a multi-resolution description of the pose for each body-part. As opposed to the previous method which analyzes the entire body together using the "full-body" codebooks, this method analyzes "body-parts" separately using separate codebooks.

For each frame of a mocap sequence, every body-part is encoded using the corresponding level-n codebook, to get a string of five codewords. Because of the quantization effect,

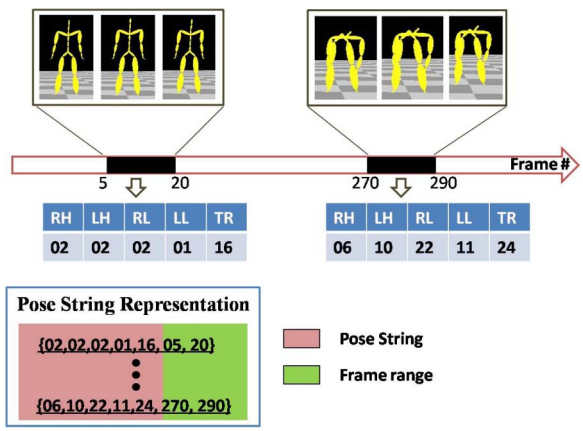


Fig. 5. A pose string representation of a mocap sequence based on body-part codewords.

these strings do not change for a chunk of consecutive frames. This chunk can be represented by a set of five body-part codeword indexes as well as the frame range through which this chunk exists. For instance, Fig. 5 shows the frames in the *bend down and pick up a box* motion sequence encoded using level-5 body-part codebooks. The black portions in the time-line indicate the frames that have the same set of five body-part poses. Three frames selected from each of the chunks are shown above the time-line, and the level-5 codeword indices are shown below, where RH, LH, RL, LL and TR represent the Right Hand, Left Hand, Right Leg, Left Leg and Torso codewords, respectively. The codeword index numbers are in the range, 0 to 31 *i.e.* (2^5-1) for each of the level-5 codebooks. All the frames in each chunk are represented collectively using a set of five body-part codeword indexes and the corresponding frame range. This compact notation is called the "pose string representation".

The pose string representation is used to obtain the body-part codeword histograms. The level-n histograms obtained for each body part (namely, right hand, left hand, right leg, left leg and the torso) for the jth motion clip are concatenated to get the level-n pose histogram vector, denoted by $\overrightarrow{x_j}$. If $\overrightarrow{h_j}^{TR}$ is the level-n torso histogram for the jth clip, $\overrightarrow{h_j}^{RH}$ is the level-n right hand histogram, and so on, then

$$\overrightarrow{x_j} = \{\overrightarrow{h_j}^{TR}, \overrightarrow{h_j}^{RH}, \overrightarrow{h_j}^{LH}, \overrightarrow{h_j}^{RL}, \overrightarrow{h_j}^{LL}\}. \quad (7)$$

The pose histograms of motions belonging to the same category are expected to be similar. The level-n pose histogram vector offers the information about the frequency of occurrence of level-n codewords for four limbs and the torso in the pose string representation of a given motion clip. Higher TSVQ levels in these histograms correspond to a higher resolution in the spatial domain. Motions belonging to the same type show similarity up to higher levels while motions of different types begin to differ at lower levels of the histogram. The number of levels can be gradually increased to get a better match between motions. However, a compromise has to be made, since similar motions will show differences if the level becomes too high. In other words, we start to see the characteristics of an individual motion clip rather than that of a motion category.

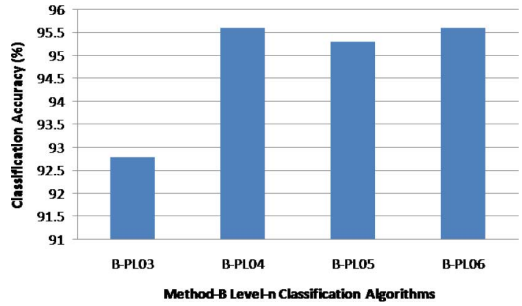


Fig. 6. Correct classification results of Method B with level-n codewords, where $n = 3; 4; 5; 6$.

Support vector machines[1] are supervised learning models for regression analysis and binary classification. Assume the classification labels $y_j \in \{+1, -1\} \forall j$; the decision rule $C(x_j) = sign(w^T \phi(x_j) + b)$; x_j to be the input feature vector for the j^{th} motion, $\phi(\cdot)$ the mapping function, classifier $C(\cdot)$ and w to be the parameter vector. Then the optimization algorithm trains the SVM classifier by minimizing the hinge loss function in form of

$$\min_{w,b} \sum_j max(0, 1 - y_j(w^T \phi(x_j) + b)) + \frac{\lambda}{2}\|w\|_2^2 \quad (8)$$

where the last term λ is the regularizer coefficient. SVM constructs a hyperplane with the largest separation between the two classes.

For multi-class classification with $\tilde{m}$ classes, $\tilde{m}(\tilde{m}-1)/2$ binary SVM classifiers are built. Every binary SVM classifier C_{ab} distinguishes between a pair of classes (a, b) where $a, b \in \{1, 2 \cdots \tilde{m}\}$ and $a \neq b$.

$$C_{ab}(x_i) = \begin{cases} +1 & \text{if } x_i \in a \text{ or} \\ -1 & \text{if } x_i \in b \end{cases} \quad (9)$$

In our case, $\tilde{m} = 30$. LIBSVM [43] trains these multiple binary SVM classifiers using the level-n pose histogram feature vectors and Gaussian RBF kernel. Refer to Eq. (7) for the interpretation of pose histogram vectors. The hyper-parameters of SVM classifiers are carefully tuned using cross validation on the training set. Regularization, careful tuning of hyper-parameters and cross validation help avoid overfitting. Every classifier then categorizes the test motion into one of two classes. The winning class gets that classifier's vote. For the kth category and the ith test motion, the total votes are calculated as

$$Vote_i^k = \sum_{b>k} \max(0, C_{kb}(x_i)) + \sum_{a<k} 1 - \max(0, C_{ak}(x_i)). \quad (10)$$

In standalone Method-B, the class that gets the maximum number of votes wins. The standalone Method-B classification results are depicted in Fig. 6. Alternatively, the votes of all classes are normalized to get $\tilde{m}$ soft decision scores (one for each category) for that test motion.

[1]http://en.wikipedia.org/wiki/Support_vector_machine

TABLE III
COMPARISON OF CLASSIFICATION RESULTS

Stage	*Features*	*Accuracy*
Method-A only	A-ML12	82.3%
	A-ML13	80.5%
	A-SL12	95.6%
	A-SL13	95.6%
Method-B only	B-PL03	92.8%
	B-PL04	95.6%
	B-PL05	95.3%
	B-PL06	95.6%
Method-C	Score Fusion	98.2%
Method-D	SVM Fusion	**99.6%**

The performance of standalone Method-B which uses the spatial domain information is presented in Section 4. The classification results at various levels are analyzed to choose the best performing levels in the multi-resolution tree histograms. These shortlisted levels are then used in the decision fusion step.

The motion string similarity (Method-A) inspects the order of the full-body poses while the pose histogram classifier (Method-B) examines the frequency of body-part poses. The two methods use completely different information to analyze the mocap data. These diverse perspectives complement each other in the fusion step to enhance the overall performance. It is worthwhile to point out that the codebooks for Method-A demand deeper levels ($n = [10, 11, 12, 13]$) in TSVQ as compared to that for Method-B ($n = [3, 4, 5, 6]$). This is because Method-A considers the full-body codebook and its level has to be deep enough to offer sufficient discriminant power. In contrast, Method-B uses specialized body-part codebooks that can reach sufficient discriminating capability with shallower levels in TSVQ.

C. Method-C: Two-Step Score Fusion

For simplicity, we use the following nomenclature to represent the selected algorithm and its parameter setting:

- *A* (or *B*): Method- *A* (or Method- *B*) classification algorithm;
- *MLn*: *M*ax-parameter with *L*evel- *n* TSVQ full-body codebooks (associated with Method-A only);
- *SLn*: *S*im-parameter with *L*evel- *n* TSVQ full-body codebooks (associated with Method-A only);
- *PLn*: *P*ose histograms with *L*evel- *n* TSVQ body-parts codebooks (associated with Method-B only).

The top two algorithms in Method-A (A-SL12, A-SL13) and Method-B (B-PL04, B-PL06) are shortlisted for fusion in Method-C. Refer to Tables III–V for details. Given the individual outcomes of these four algorithms and their soft scores, Method-C makes the final decision on the motion category using a two-step procedure as outlined below.

Step 1) *Hard Decision Fusion* The individual outcomes of the above-mentioned four algorithms (for a certain test motion) are taken together. If these outcomes are biased towards a specific category, then the test motion is classified into that category. Otherwise, we move on to the second step. For example, if algorithms A-SL12, B-PL05 and B-PL06 classify the motion into *category-M* while A-SL13 classifies the motion into *category-N*. Then, Method-C classifies the motion into *category-M* since this category has the maximum number of votes. If there is any tie of some sorts, we proceed to the second step to break the tie.

Step 2) *Soft Score Fusion* If there is a tie, soft scores of all these four algorithms are merged together as

$$(A)^k_{i,Soft} = \sum_{n\in\{12,13\}} (A-SLn)^k_{i,Soft}$$
$$(B)^k_{i,Soft} = \sum_{n\in\{4,6\}} (B-PLn)^k_{i,Soft} \quad (11)$$

where $k \in \{1, 2\cdots\tilde{m}\}$ and i denote a motion category and a test motion sequence, respectively, and subscript *Soft* indicates soft scores. The category getting the maximum score is assigned to that of test motion i; namely,

$$C_i = \max_{k\in\{1,2\cdots\tilde{m}\}} \{(A)^k_{i,Soft} + w * (B)^k_{i,Soft}\} \quad (12)$$

where weight w can be determined heuristically by analyzing the variance of the soft scores and the standalone classification outcomes. For simplicity, w is set to 1 in our experiments. The performance results of Method-C are given in Table VI.

D. Method-D: Two-Step SVM Fusion

To improve the performance further, we developed another two step fusion approach based on SVM classifiers, *a.k.a.* Method-D. The details of this two step approach are given below.

Step 1) *Hard Decision Fusion* Similar to Method-C, the individual outcomes of four algorithms A-SL12, A-SL13, B-PL04 and B-PL06 (for a certain test motion) are taken together. If these outcomes are biased towards a specific category, the test motion is classified into that category. If there is a tie, we proceed to the second step.

Step 2) *SVM Fusion* For Method-D, we adopt a new mechanism to compute soft scores of Method-A and Method-B. For Method-A, we use the normalized SIM parameter values as the soft scores. For Method-B, we use the 'one-versus-rest' SVM classifier

$$C_k\{x_i\} = w^T\phi(x_i) + b \quad (13)$$

where $\phi(\cdot)$ is the mapping function, b is a constant and $k \in \{1, 2\cdots\tilde{m}\}$, to yield the soft score for feature vector x_i with respect to category k. A higher absolute positive score implies a higher possibility of test motion i belonging to class k while a higher absolute negative scores imply otherwise.

The soft scores, computed in this fashion for the four aforementioned algorithms *i.e.* A-SL12, A-SL13, B-PL04 and B-PL06, are combined to form the following four dimensional feature vectors:

$$\overrightarrow{p^k_i} = \{(A-SL12)^k_{i,Soft}, (A-SL13)^k_{i,Soft}, (B-PL04)^k_{i,Soft}, (B-PL06)^k_{i,Soft}\} \quad (14)$$

TABLE IV
STANDALONE MOTION STRING SIMILARITY COMPARISON RESULTS

Sr.No.	Category(Motions)	A − ML12	A − ML13	A − SL12	A − SL13
1	Run(27)	26(96%)	26(96%)	27(100%)	26(96%)
2	Walk(47)	40(85%)	40(85%)	46(97%)	47(100%)
3	Forward Jump(9)	8(88%)	8(88%)	8(88%)	9(100%)
4	Forward Dribble(5)	5(100%)	5(100%)	5(100%)	5(100%)
5	Cartwheel(5)	5(100%)	5(100%)	5(100%)	5(100%)
6	Kickball(6)	6(100%)	6(100%)	6(100%)	6(100%)
7	Boxing(7)	0(0%)	0(0%)	6(85%)	7(100%)
8	Mickey Walk(7)	7(100%)	7(100%)	7(100%)	7(100%)
9	Sit and Stand(5)	4(80%)	4(80%)	5(100%)	5(100%)
10	Laugh(6)	4(66%)	6(100%)	6(100%)	6(100%)
11	Sweep Floor(5)	2(40%)	2(40%)	5(100%)	5(100%)
12	Wash Windows(5)	3(60%)	3(60%)	5(100%)	5(100%)
13	Climb Ladder(5)	5(100%)	5(100%)	5(100%)	5(100%)
14	Steps(7)	4(57%)	6(85%)	7(100%)	7(100%)
15	Eating(5)	5(100%)	5(100%)	5(100%)	5(100%)
16	Tiptoe(5)	5(100%)	3(60%)	5(100%)	5(100%)
17	Pick Box/Bend Waist(6)	6(100%)	6(100%)	5(83%)	6(100%)
18	Limp(5)	5(100%)	4(80%)	5(100%)	5(100%)
19	Balancing Walk(12)	10(83%)	9(75%)	12(100%)	12(100%)
20	Get Up From Chair(5)	4(80%)	4(80%)	5(100%)	4(80%)
21	Breast Stroke(6)	3(50%)	1(16%)	5(83%)	5(83%)
22	Hop on Left Foot(6)	4(66%)	6(100%)	6(100%)	6(100%)
23	Bouncy Walk(6)	4(66%)	4(66%)	6(100%)	6(100%)
24	Marching(10)	10(100%)	10(100%)	10(100%)	9(90%)
25	Rhyme Tea Pot(16)	13(81%)	13(81%)	12(75%)	12(75%)
26	Rhyme Cock Robin(15)	9(60%)	6(40%)	13(86%)	13(86%)
27	Swing(10)	10(100%)	10(100%)	9(90%)	10(100%)
28	Placing Tee(5)	5(100%)	5(100%)	5(100%)	4(80%)
29	Salsa Dance(15)	13(86%)	11(73%)	15(100%)	14(95%)
30	Get Up From Floor(5)	4(80%)	4(80%)	5(100%)	5(100%)
	Total(278)	**229(82.3%)**	**224(80.5%)**	**266(95.6%)**	**266(95.6%)**

TABLE V
STANDALONE POSE HISTOGRAM CLASSIFICATION RESULTS

Sr.No.	Category(Motions)	B − PL03	B − PL04	B − PL05	B − PL06
1	Run(27)	26(96%)	27(100%)	26(96%)	26(96%)
2	Walk(47)	46(97%)	46(97%)	46(97%)	46(97%)
3	Forward Jump(9)	8(88%)	8(88%)	7(77%)	9(100%)
4	Forward Dribble(5)	5(100%)	5(100%)	5(100%)	5(100%)
5	Cartwheel(5)	5(100%)	5(100%)	5(100%)	5(100%)
6	Kickball(6)	5(83%)	5(83%)	5(83%)	5(83%)
7	Boxing(7)	7(100%)	7(100%)	7(100%)	7(100%)
8	Mickey Walk(7)	6(85%)	7(100%)	7(100%)	7(100%)
9	Sit and Stand(5)	5(100%)	5(100%)	5(100%)	4(80%)
10	Laugh(6)	6(100%)	6(100%)	6(100%)	6(100%)
11	Sweep Floor(5)	4(80%)	5(100%)	5(100%)	5(100%)
12	Wash Windows(5)	5(100%)	5(100%)	5(100%)	5(100%)
13	Climb Ladder(5)	5(100%)	5(100%)	5(100%)	5(100%)
14	Steps(7)	7(100%)	7(100%)	7(100%)	7(100%)
15	Eating(5)	5(100%)	5(100%)	5(100%)	5(100%)
16	Tiptoe(5)	5(100%)	5(100%)	5(100%)	5(100%)
17	Pick Box/Bend Waist(6)	5(83%)	5(83%)	6(100%)	6(100%)
18	Limp(5)	5(100%)	5(100%)	5(100%)	5(100%)
19	Balancing Walk(12)	10(83%)	12(100%)	11(91%)	11(91%)
20	Get Up From Chair(5)	4(80%)	5(100%)	5(100%)	3(60%)
21	Breast Stroke(6)	6(100%)	6(100%)	5(83%)	5(83%)
22	Hop on Left Foot(6)	6(100%)	5(83%)	6(100%)	6(100%)
23	Bouncy Walk(6)	3(50%)	4(66%)	5(83%)	6(100%)
24	Marching(10)	10(100%)	10(100%)	10(100%)	10(100%)
25	Rhyme Tea Pot(16)	12(75%)	14(87%)	14(87%)	14(87%)
26	Rhyme Cock Robin(15)	14(93%)	13(86%)	12(80%)	15(100%)
27	Swing(10)	10(100%)	10(100%)	10(100%)	10(100%)
28	Placing Tee(5)	4(80%)	4(80%)	5(100%)	4(80%)
29	Salsa Dance(15)	15(100%)	15(100%)	15(100%)	15(100%)
30	Get Up From Floor(5)	4(80%)	5(100%)	5(100%)	4(80%)
	Total(278)	**258(92.8%)**	**266(95.6%)**	**265(95.3%)**	**266(95.6%)**

Suppose j is the training motion. In the training phase, feature vectors p_j^k belonging to class k are used to build new 'one-versus-rest' SVM classifiers N_k for class k. In the testing phase, the four dimensional feature vectors p_i^k and the corresponding

TABLE VI
DECISION FUSION RESULTS

Sr.No.	Category(Motions)	Method – C	Method – D
1	Run(27)	27(100%)	27(100%)
2	Walk(47)	47(100%)	47(100%)
3	Forward Jump(9)	9(100%)	9(100%)
4	Forward Dribble(5)	5(100%)	5(100%)
5	Cartwheel(5)	5(100%)	5(100%)
6	Kickball(6)	6(100%)	6(100%)
7	Boxing(7)	7(100%)	7(100%)
8	Mickey Walk(7)	7(100%)	7(100%)
9	Sit and Stand(5)	5(100%)	5(100%)
10	Laugh(6)	6(100%)	6(100%)
11	Sweep Floor(5)	5(100%)	5(100%)
12	Wash Windows(5)	5(100%)	5(100%)
13	Climb Ladder(5)	5(100%)	5(100%)
14	Steps(7)	7(100%)	7(100%)
15	Eating(5)	5(100%)	5(100%)
16	Tiptoe(5)	5(100%)	5(100%)
17	Pick Box/Bend Waist(6)	6(100%)	6(100%)
18	Limp(5)	5(100%)	5(100%)
19	Balancing Walk(12)	12(100%)	12(100%)
20	Get Up From Chair(5)	5(100%)	5(100%)
21	Breast Stroke(6)	5(83%)	6(100%)
22	Hop on Left Foot(6)	6(100%)	6(100%)
23	Bouncy Walk(6)	6(100%)	6(100%)
24	Marching(10)	10(100%)	10(100%)
25	Rhyme Tea Pot(16)	12(75%)	15(93%)
26	Rhyme Cock Robin(15)	15(100%)	15(100%)
27	Swing(10)	10(100%)	10(100%)
28	Placing Tee(5)	5(100%)	5(100%)
29	Salsa Dance(15)	15(100%)	15(100%)
30	Get Up From Floor(5)	5(100%)	5(100%)
	Total(278)	**273(98.2%)**	**277(99.6%)**

classifiers N_k are used to find the SVM scores $N_k\{p_i^k\}$ for i and its corresponding class k. Then, in Method-D, the category having the maximum SVM score, *i.e.*,

$$D_i = \max_{k\in\{1,2\cdots\bar{m}\}} \{N_k\{x_i\}\} \tag{15}$$

is the category of test motion i. The performance results of Method-D are presented in Table VI.

IV. EXPERIMENTAL RESULTS

A. Dataset Description

Our experimental dataset contains a total of 278 high quality labeled motion clips ($\sim$0.5 million frames) belonging to 30 distinct categories taken from the CMU mocap database [5]. 33 different subjects were involved in recording these motions. The captured data has 62 dimensions of freedom (DOFs). The first three dimensions give the 3D position of the centroid of the human skeleton while the remaining 59 DOFs describe the angular positions of the joints as shown in Table I. These 59 DOFs together define a full-body pose in the 3D space. The pre-processing step irons out the kinks in the full-body raw data file and also separates it into five body-part files for Method-B. The generated TSVQ codebooks are used to classify the motions from the dataset using n-fold cross validation process.

The n-fold cross validation method is adopted to evaluate the performance of the proposed classification algorithms. In this procedure, all motion clips of the same category are divided into n subsets. We choose an arbitrary subset as the test data and the other $n-1$ subsets as the training data, and conduct the experiment to get the classification performance. Following the same procedure, we can perform n such tests using each subset as a test subset once. Finally, the classification results for all n tests are aggregated together. We set $n = 5$ in our experiment. Fig. 7 shows the cross validation results for each of the five groups.

B. Classification Performance and Discussion

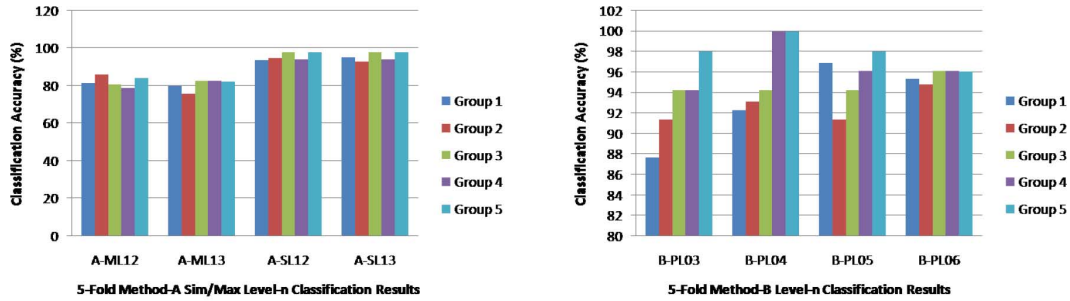
We compared the performance of the four classification methods: 1) Standalone Method-A, 2) Standalone Method-B, 3) Method-C and 4) Method-D. The overall correct classification results for 278 test motions using the 5-fold cross validation with different level TSVQ codebooks and parameter settings are summarized in Table III. More detailed classification results are listed in Tables IV–VI.

In contrast with the pose histogram classification approach (Method-B), the motion string similarity comparison approach (Method-A) examines the transition of "coarsely quantized poses" in the temporal domain. Though there is an approximation, the temporal information helps in classifying some of the motions correctly, which the pose histograms missed. For instance, the misclassified *Kickball* motions are all correctly categorized by the temporal domain approach, indicating the fact that the sequence of poses do matter. Please check A-SL12 and B-PL06 columns in Tables IV and V, respectively. In addition, the temporal domain approach reinforces the confidence in the correctly classified motions. On the flip side, due to the coarse quantization used in the temporal domain approach, a lot of motions from the *Cock Robin* category are misclassified as shown in column A-SL13, Table IV. However, the finer resolution of the pose histogram classification algorithms give us correct results for those motions as shown in column B-PL06 of Table V. In this manner, these two diverse approaches complement each other.

Both the temporal and the spatial domain approaches have their own strengths and shortcomings. Due to this diversity, it is desired to combine them so as to improve the classification accuracy even further. Both the fusion approaches are able to do so. For example, the *Run* and the *Walk* motions which were not classified by either Method-A or Method-B are correctly classified by both the fusion approaches. This demonstrates the significance of proper fusion. Within these two fusion approaches, the SVM fusion (Method-D) outperforms the score fusion (Method-C). This is mainly due to the fact that Method-C weighs all the features equally while Method-D is able to put more emphasis on the relevant features. This effect is more pronounced for the *Tea Pot* nursery rhyme category as shown in Table VI.

The performance results of the state-of-the-art mocap classification algorithms, tested on similar datasets, are presented in Table VII. Our classification results are better than the other. However, the direct comparison of all these algorithms with ours is difficult due to the following reasons.

- Most of state-of-the-art algorithms tested their approach on their own datasets (or a mixture of CMU and self-generated datasets). We have no access to their datasets. Our ap-



(a) (b)

Fig. 7. (a) Group-wise performance using the sim and max parameters and (b) group-wise classification results for pose histogram classifiers.

TABLE VII
PERFORMANCE COMPARISON OF HUMAN MOCAP DATA CLASSIFICATION ALGORITHMS

Algorithms	*Accuracy*	*Dataset*
D-SBGPLVM [45]	72.9%	CMU dataset
EROS [4]	87.5% [1]	Self generated
kWAS [26]	90.3% [1]	Self generated
Wu *et al.* [31]	97.0% [1]	CMU + Self generated
Kadu *et al.* [6]	97.0% [2]	CMU dataset
Wu *et al.* [32]	98.1% [1]	CMU + Self generated
Proposed Method	**99.6%**	CMU dataset

proach uses the motions from the freely available online CMU mocap database [5] only as listed in the Appendix.

- We adopted a stricter classification performance evaluation metric. For example, Wu *et al.* [32] clustered the *running* and *marching* motions into one group as these motions are similar. In contrast, our evaluation metric requires that, if the labels of motions are different, they should be categorized into separate groups. Apart from that, some motions belonging to the same category are clustered into separate sub-groups in [32]. As long as samples in each sub-group are homogeneous, it is treated as correct clustering. However, they are treated as misclassification in our performance metric.
- We have more motion categories as compared to others. For example, our dataset contains both simple and complex motions from 30 categories while Wu's dataset had only simple motions from 14 categories.

Although we do not have a direct performance comparison because of different datasets, evaluation metrics and action categories used. The results in Table VII still offer a rough idea of the performance of prior art.

V. CONCLUSION AND FUTURE WORK

A technique for automated mocap data classification was presented in this work. The TSVQ method was adopted to approximate static human poses with codewords while a dynamic human motion was represented by a sequence of codewords. Fusion approaches were proposed to classify mocap data into different categories. We tested the proposed algorithms on the CMU mocap database using the 5-fold cross validation procedure and obtained a correct classification rate of 99.6%.

The proposed algorithm can be extended to various tasks required by mocap database management such as segmentation, indexing and retrieval, without much effort. We only sketch the basic ideas below.

- *Segmentation*: If a complex motion clip contains multiple basic motions, then the sequence of these basic motions will be present in the sequence of that complex motion. By using string matching, we can find out the location of sequences of all basic motions in the complex motion. This corresponds to the segmentation of a complex motion sequence into multiple basic motion sequences.
- *Indexing*: After running the classification algorithm, unknown motion clips can be classified into one of the known categories and indexed accordingly. Consider a mixed motion that has running, jumping and bending actions one after the other. Using the string matching and the database of sequences, the location of all these basic motions in that mixed motion can be determined and these basic motions can be indexed accordingly.
- *Retrieval*: The retrieval problem can be solved using suffix array technique. To be more specific, for a given test motion, we would like to extract motion clips or parts of motion clips from the database that are similar to the query. To perform this retrieval task, we convert the test motion into a sequence and use the string matching algorithm to search the motion clips or portions of the motion clips that have the same sequence. Those clips can then be retrieved.

There are several algorithms for extracting human skeleton from depth images produced by Microsoft Kinect [45], [46]. The data generated by these algorithms are similar to the mocap data [47]. It is interesting to explore whether the proposed methodology can be extended to the Kinect sensor data as well. High quality 3D rendering can be achieved by blending stored mocap data with the pose information from the Kinect sensor inputs.

APPENDIX

The CMU mocap database [5] motion capture files used in our research, are listed below. All these files are $.amc$ files containing the frame by frame values of 59 rotational angles of the 29 joints in the human skeleton, plus a triplet representing the 3D position of the root joint at that moment. The number before the '_' symbol is the subject identification number and the one after it is the motion serial number. For example, 09_01 is the $09\_01.amc$ file which is the first motion of the subject '09'. The corresponding skeleton file having the details of the subject's skeleton structure is $09.asf$.

1) *Run*: 09_01, 09_02, 09_03, 09_04, 09_05, 09_06, 09_07, 09_08, 09_09, 09_10, 09_11, 35_17, 35_18, 35_19, 35_20, 35_21, 35_22, 35_23, 35_24, 35_25, 35_26, 127_06, 127_07, 127_08, 141_01, 141_02, 16_55.
2) *Walk*: 35_01, 35_02, 35_03, 35_04, 35_05, 35_06, 35_07, 35_08, 35_09, 35_10, 35_11, 35_12, 35_13, 35_14, 35_15, 35_16, 35_28, 35_29, 35_30, 35_31, 02_01, 02_02, 07_01, 07_02, 07_03, 07_06, 07_07, 07_08, 07_09, 07_10, 07_11, 08_01, 08_02, 08_03, 08_06, 08_08, 08_09, 08_10, 12_01, 12_02, 12_03, 16_15, 16_16, 16_21, 16_22, 16_31, 16_32.
3) *Forward Jump*: 16_05, 16_06, 16_07, 16_09, 16_10, 13_11, 13_13, 13_19, 13_32.
4) *Forward Dribble*: 06_02, 06_03, 06_05, 06_10, 06_11.
5) *Cartwheel*: 49_06, 49_07, 90_02, 90_03, 90_04.
6) *Kickball*: 10_01, 10_02, 10_03, 10_05, 10_06.
7) *Boxing*: 13_17, 13_18, 14_01, 14_02, 14_03, 15_13, 17_10.
8) *Mickey Walk*: 120_08, 120_09, 120_10, 120_11, 120_12, 120_13, 120_14.
9) *Sit and Stand Up*: 13_01, 13_02, 13_03, 14_27, 14_28.
10) *Laugh*: 13_14, 13_15, 13_16, 14_17, 14_18, 14_19.
11) *Sweep Floor*: 13_23, 13_24, 13_25, 14_16 (2).
12) *Wash Windows*: 13_20, 13_21, 13_22, 14_10, 14_11.
13) *Climb Ladder*: 13_33, 13_34, 14_33, 14_34, 14_35.
14) *Steps*: 13_35, 13_36, 13_37, 13_38, 14_21, 14_22, 14_23.
15) *Eating*: 79_12, 79_15, 79_42, 80_24, 80_33.
16) *Tiptoe*: 13_10, 13_12, 14_07, 14_08, 14_09.
17) *Pick Box Bend Waist*: 115_01, 115_02, 115_03, 115_04, 115_05, 115_10.
18) *Limp*: 77_19, 77_20, 77_22, 77_23, 77_24.
19) *Balance*: 132_01, 132_02, 132_03, 132_04, 132_05, 132_06, 132_07, 132_08, 132_09, 132_10, 132_11, 132_12.
20) *Get Up From Chair*: 111_09, 111_10, 111_11, 114_04, 114_06.
21) *Breast Stroke*: 125_01, 125_02, 125_04, 126_03, 126_04, 126_05.
22) *Hop on Left Foot*: 132_23, 132_24, 132_25, 132_26, 132_27, 132_28.
23) *Bouncy Walk*: 132_29, 132_30, 132_31, 132_32, 132_33, 132_34.
24) *Marching*: 138_01, 138_02, 138_03, 138_04, 138_05, 138_06, 138_07, 138_08, 138_09, 138_10.
25) *Nursery Rhyme Tea Pot*: 24_01, 25_01, 26_03, 26_04, 27_03, 27_07, 28_02, 28_06, 29_03, 29_08, 30_02, 30_08, 31_02, 31_06, 32_04, 32_08.
26) *Nursery Rhyme Cock Robin*: 26_07, 26_08, 27_05, 27_10, 28_04, 28_08, 29_06, 29_10, 30_05, 30_06, 30_10, 31_04, 31_08, 32_06, 32_10.
27) *Swing*: 64_01, 64_02, 64_03, 64_04, 64_05, 64_06, 64_07, 64_08, 64_09, 64_10.
28) *Placing Tee*: 64_16, 64_17, 64_18, 64_19, 64_20.
29) *Salsa Dance*: 60_01, 60_02, 60_03, 60_04, 60_05, 60_06, 60_07, 60_08, 60_09, 60_10, 60_11, 60_12, 60_13, 60_14, 60_15.
30) *Get Up From Floor*: 111_06, 111_07, 111_08, 140_08, 140_09.

REFERENCES

[1] B.-S. Chew, L.-P. Chau, and K.-H. Yap, "A fuzzy clustering algorithm for virtual character animation representation," *IEEE Trans. Multimedia*, vol. 13, no. 1, pp. 40–49, Feb. 2011.

[2] W. Schollhorn, D. Stefanyshyn, and W. Liu, "Identification of individual walking patterns using time discrete and time continuous data sets," *Gait Posture*, vol. 15, pp. 180–186, 2002.

[3] C. Li, P. Kulkarni, L. Liu, B. Prabhakaran, and L. Khan, "Real-time classification of multivariate motion data using support vector machines (SVM)," in *Proc. 5th Int. Workshop Multimedia Data Mining*, 2004, pp. 1–7.

[4] K. Yang and Shahabi, "A PCA-based similarity measure for multivariate time series," in *Proc. 2nd ACM Int. Workshop Multimedia Databases*, 2004, pp. 65–74.

[5] "Carnegie-Mellon mocap database," Carnegie Mellon Univ. Pittsburgh, PA, USA, Mar. 2007 [Online]. Available: http://mocap.cs.cmu.edu/

[6] H. Kadu, M. Kuo, and C.-C. J. Kuo, "Human motion classification and management based on mocap data analysis," in *Proc. Joint ACM Workshop Human Gesture Behavior Understanding*, Dec. 2011, pp. 73–74.

[7] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large data sets," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 559–568, Aug. 2004.

[8] A. Safonova, J. Hodgins, and N. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 514–521, Aug. 2004.

[9] L. Ren, G. Shakhnarovich, J. Hodgins, H. Pfister, and P. Viola, "Learning silhouette features for control of human motion," *ACM Trans. Graphics*, vol. 24, no. 4, pp. 1303–1331, Oct. 2005.

[10] J. Lee, J. Chai, P. Reitdma, J. Hodgins, and N. Pollard, "Interactive control of avatars animated with human motion data," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 491–500, 2002.

[11] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast, "Dynamic response for motion capture animation," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 697–701, 2005.

[12] K. Pullen and C. Bregler, "Motion capture assisted animation: Texturing and synthesis," in *Proc. ACM SIGGRAPH*, 2002, pp. 501–508.

[13] K. Liu and Z. Popović, "Synthesis of complex dynamic character motion from simple animations," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 408–416, Jul. 2002.

[14] K. Forbes and E. Fiume, "An efficient search algorithm for motion data using weighted PCA," in *Proc. Eurograph./ACM SIGGRAPH Symp. Comput. Animation*, Jul. 2005, pp. 67–76.

[15] F. Liu, Y. Zhuan, F. Wu, and Y. Pan, "3d motion retrieval with motoin index tree," *Comput. Vis. Image Understanding*, vol. 92, no. 2–3, pp. 265–284, 2003.

[16] Y. Sakamoto, S. Kuriyama, and T. Ka, "Motion map: Image-based retrieval and segmentation of motion data," in *Proc. Eurograph./ACM SIGGRAPH Symp. Comput. Animation*, Jul. 2004, pp. 259–266.

[17] C. Chiu, S. Chao, M. Wu, S. Yang, and H. Lin, "Content-based retrieval for human motion data," *J. Vis. Commun. Image Representation*, vol. 15, no. 3, pp. 446–466, 2004.

[18] M. Y. Wu, S. Chao, S. Yang, and H. Lin, "Content-based retrieval for human motion data," in *Proc. IPPR Conf. Comput. Vis., Graphics Image Process.*, 2003, pp. 605–612.

[19] E. Hsu, K. Pulli, and J. Popović, "Style translation for human motion," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 1082–1089, Jul. 2005.

[20] O. Arikan and D. A. Forsythe, "Interactive motion generation from examples," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 483–490, 2002.

[21] T. Kim, S. Park, and S. Shin, "Rhythmic-motion synthesis base on motionbeat analysis," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 392–401, 2003.

[22] L. Kovar and M. Gleicher, "Flexible automatic motion blending with registration curves," in *Proc. Eurograph./ACM SIGGRAPH Symp. Comput. Animation*, 2003, pp. 214–224.

[23] J. Wang and B. Bodenheimer, "An evaluation of a cost metric for selecting transitions between motion segments," in *Proc. Eurograph./ACM SIGGRAPH Symp. Comput. Animation*, 2003, pp. 232–238.

[24] O. Arikan, D. A. Forsythe, and O. Brien, "Motion synthesis from annotations," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 402–408, 2003.

[25] M. Cardle, M. Vlachos, S. Brooks, E. Keogh, and D. Gunopulos, "Fast motion capture matching with replicated motion editing," in *Proc. SIGGRAPH Tech. Sketches Appl.*, 2003.

[26] C. Li, S. Q. Zheng, and B. Prabhakaran, "Segmentation and recognition of motion streams by similarity search," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 3, no. 16, p. 16, 2007.
[27] M. Müller, T. Röder, and M. Clausen, "Efficient indexing and retrieval of motion capture data based on adaptive segmentation," in *Proc. 4th Int. Workshop Content-Based Multimedia Indexing*, 2005, pp. 677–685.
[28] M. Müller, T. Röder, and M. Clausen, "Efficient content-based retrieval of motion capture data," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 677–685, 2005.
[29] M. Müller and T. Röder, "Motion template for automatic classification and retrieval of motion capture data," in *Proc. Eurograph./ACM SIGGRAPH Symp. Comput. Animation*, 2006, pp. 137–146.
[30] B. Rosenhahn, R. Klette, and D. Metaxas, *Human Motion: Understanding, Modeling, Capture and Animation*. Dordrecht, The Netherlands: Springer, 2008.
[31] S. Wu, S. Xia, Z. Wang, and C. Li, "Efficient motion data indexing and retrieval with local similarity measure of motion strings," *Vis. Comput.*, vol. 25, no. 5–7, pp. 499–508, 2009.
[32] S. Wu, Z. Wang, and S. Xia, "Indexing and retrieval of human motion data by a hierarchical tree," in *Proc. VRST*, Nov. 2009, pp. 207–214.
[33] S. Sadanand and J. Corso, "Action bank: A high-level representation of activity in video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1234–1241.
[34] R. Kishore and M. Shah, "Recognizing 50 human action categories of web videos," *Mach. Vis. Appl.*, vol. 24, no. 5, pp. 971–981, 2013.
[35] M. Kuo, P.-Y. Chiang, and C.-C. J. Kuo, "Overview on mocap data compression," in *Proc. APSIPA Annu. Summit Conf.*, Biopolis, Singapore, Dec. 2010, pp. 853–858.
[36] M. C. Kuo, P.-Y. Chiang, and C.-C. J. Kuo, "Coding of motion capture data via temporal-domain sampling and spatial-domain vector quantization techniques," in *Proc. Pacific-Rim Conf. Multimedia*, Shanghai, China, Sep. 2010, pp. 84–99.
[37] M. Effros, "Practical multi-resolution source coding: TSVQ revisited," in *Proc. Data Compression Conf.*, 1998, pp. 53–62.
[38] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.*, vol. 36, no. 8, pp. 957–971, Aug. 1988.
[39] M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch, "Replacing suffix trees with enhanced suffix arrays," *J. Discrete Algorithms*, vol. 2, no. 1, pp. 53–86, 2004.
[40] J. Karkkainen and P. Sanders, "Simple linear work suffix array construction," in *Proc. 13th Int. Conf. Automata, Languages Program.*, 2003, pp. 943–955.
[41] T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park, "Linear-time longest-common-prefix computation in suffix arrays and its applications," in *Proc. 12th Annu. Symp. Combinatorial Pattern Matching*, London, U.K., 2001, pp. 181–192.
[42] D. K. Kim, J. S. Sim, H. Park, and K. Park, "Constructing suffix arrays in linear time," *J. Discrete Algorithms*, vol. 3, no. 2, pp. 126–142, 2005.
[43] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011.
[44] V. Ntouskos, P. Papadakis, and F. Pirri *et al.*, "Discriminative sequence back-constrained GP-LVM for mocap based action recognition," in *Proc. Int. Conf. Pattern Recog. Appl. Methods*, 2013.
[45] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, no. 1, pp. 116–124, 2013.
[46] D. S. Alexiadis and P. Daras, "Quaternionic signal processing techniques for automatic evaluation of dance performances from mocap data," *IEEE Trans. Multimedia*, vol. 16, no. 5, pp. 1391–1406, Aug. 2013.
[47] X. Chen and M. Koskela, "Classification of rgb-d and motion capture sequences using extreme learning machine," in *Image Anal.*, 2013, pp. 640–651.

Harshad Kadu received the Bachelor's degree in electronics and communication engineering from Visvesvaraya National Institute of Technology, Nagpur, India, in 2008, the Master's degree from the Viterbi School of Engineering, University of Southern California (USC), Los Angeles, CA, USA, in 2011, and is currently working towards the Ph.D. degree with the Ming Hsieh Department of Electrical Engineering, USC, Los Angeles, CA, USA.

His research interests include motion capture data, mocap database management, and realistic 3D human motion synthesis.

C.-C. Jay Kuo (S'83–M'86–SM'92–F'99) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1980, and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1985 and 1987, respectively.

He is currently the Director of the Media Communications Lab and Dean's Professor of Electrical Engineering at the University of Southern California, Los Angeles, CA, USA. His research interests include digital image/video analysis and multimedia data compression.

Dr. Kuo is a Fellow of the American Association for the Advancement of Science (AAAS) and the International Society for Optical Engineers (SPIE). He is the President of the Asia-Pacific Signal and Information Processing Association (APSIPA).