Indexing of Motion Capture Data for Efficient and Fast Similarity Search

Chuanjun Li
Department of Computer Science
The University of Texas at Dallas, Richardson, TX, USA
Email: chuanjun@utdallas.edu

B. Prabhakaran
Department of Computer Science
The University of Texas at Dallas, Richardson, TX, USA
Email: praba@utdallas.edu

***Abstract*— As motion capture systems are increasingly used for motion tracking and capture, and more and more surveillance cameras are installed for security protection, more and more motion data, including 3D motion data becomes available, making it important to index motion data for quick retrieval of similar motions. This paper proposes a simple and efficient tree structure for indexing motion data with dozens of attributes. Feature vectors are extracted for indexing by using singular value decomposition (SVD) properties of motion data matrices. By having similar motions with large variations indexed together, searching for similar motions of a query needs only one node traversal at each tree level, and only one feature needs to be considered at one tree level. Experiments with real hand gestures, arm motions and full body motions show that the majority of irrelevant motions can be pruned while retrieving all similar motions, and the traversal of an indexing tree for a query takes only a few microseconds.**

***Index Terms*— Motion capture, singular value decomposition, indexing**

I. Introduction

With the increasing use of motion capture devices and surveillance cameras in various application domains including life sciences, animations and security, more and more multi-attribute motion data sets are becoming available, making it increasingly necessary to develop efficient and fast indexing approaches for such data. An indexing structure or pruning algorithm should prune most of the irrelevant motions in a large motion database for a motion query in real time while retrieving all or almost all similar ones.

To prune motions efficiently and fast needs to address several challenges as indicated in Figure 1 for two similar full body motions:

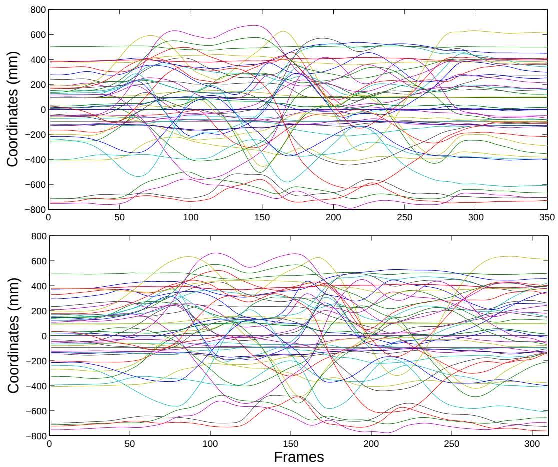

Figure 1. Similar full body motions of different lengths

- Datasets of motions have dozens of attributes. To capture a complete human motion, dozens of body segment joints need to be considered. To describe the angular values or coordinates of the joints, dozens of attributes are needed accordingly.
- Datasets of motions have different lengths. Motions can have different durations and can be carried out at different speeds, causing motion datasets to have different lengths.
- Datasets of motions can have both local scaling and global scaling due to differences in motion speeds at different times.

Due to these issues, direct indexing of motion data is difficult and inefficient.

This paper proposes a new method for indexing motion data with dozens of attributes. Instead of indexing the multi-attribute data matrices of different lengths, we extract feature vectors of equal dimensions from the data matrices and then index the feature vectors. The feature vectors are extracted by obtaining dominating vectors from singular value decompositions (SVD) of motion data and by reducing vector dimensionalities. Corresponding

This paper is based on “A Novel Indexing Approach for Efficient and Fast Similarity Search of Captured Motions ,” by Chuanjun Li and B. Prabhakaran, which appeared in the Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Singapore, April 2006.

This work was supported in part by the National Science Foundation under Grant No. 0237954 for the project CAREER: Animation Databases and by the US Army Research Office grant 48645-MA on ”Storage, Retrieval, and Delivery of 3D Models and Multi-attribute Motion Data”

feature values of all motion patterns are partitioned into several intervals. Motion or feature vector identifiers (ID) are inserted into a tree of feature intervals by using the corresponding feature values. To take into consideration motion variations, a feature ID is allowed to be inserted into multiple neighboring feature intervals. Hence a feature vector ID can be in multiple leaf nodes instead of in only one leaf node.

This paper has the following contributions:

- The proposed indexing technique is applicable to data with dozens of attributes and with different lengths, and data can have local and global scaling.
- Pruning efficiency can be as high as 95% for real data.
- The traversal of an index tree for a query takes only a few microseconds.

II. Related Work

Indexing has been attempted for multi-attribute data from many application fields by different techniques. Equal length multi-attribute sequences were considered in [1]. A CS-Index structure was proposed for shift and scale transformations. In [2], multi-attribute sequences were partitioned into subsequences, each of which was contained in a Minimum Bounding Rectangle (MBR). Every MBR was indexed and stored into a database by using an R-tree or any of its variants. Dynamic time warping (DTW) and longest common subsequence (LCSS) were extended for similarity measures of multi-attribute data in [3]. Before the exact LCSS or DTW was performed, sequences were segmented into MBRs to be stored in an R-tree. Based on the MBR intersections, similarity estimates were computed to prune irrelevant sequences.

Attributes of the data indexed in the previous work are less than ten. In contrast, our proposed indexing structure can handle dozens or hundreds of data attributes without loss of good performances. This work proposes a novel indexing approach which is different from that in [4], making it possible to search an indexing tree for similar motions in only several microseconds.

Lower bounding has been proposed in [5], [6] for pruning multi-attribute motions. Global scaling was considered by uniformly stretching a shorter time series to match the corresponding prefix of a longer time series in [5], and search was sped up by lower envelops. Similarly, lower bounds were used to prune irrelevant data at each level of a multilevel distance-based index structure [6]. In contrast, this paper extracts feature vectors and indexes the feature vectors by an interval-based index tree.

Posture-based indexing techniques have been developed in [7]–[9]. [7] constructed qualitative features describing geometric relations between specified body points of a pose, and used these features to induce a time segmentation of motion capture data streams for motion indexing. [8] used a hierarchical motion description for a posture, and used key frame extraction for retrieving motions. Motion poses were clustered in [9] using piecewise-linear models and indexing structures were constructed

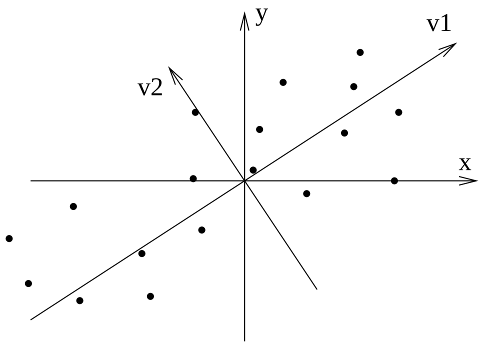

Figure 2. Geometric structure of an 18×2 matrix exposed by its SVD

according to the transition trajectories through these linear components. In contrast, our proposed indexing technique does not considered specific poses, and can be applicable to any multi-attribute data sequences.

III. Geometric Structures Revealed by SVD

In this section, we give the definition and geometric interpolation of SVD for its application to the indexing of multi-attribute motion data.

SVD exposes the geometric structure of a matrix A. If the n-dimensional row vectors or points in A have different variances along different directions in the n-dimensional space, and columns of A have zero means, the SVD of matrix A can find the direction with the largest variance. If columns of A do not have zero means, the direction along which row vector projections have the largest 2-norm or Euclidean length can be revealed by SVD. Figure 2 illustrates the data in an 18×2 matrix. The 18 points in the 18×2 matrix have different variances along different directions, and the points have the largest variance along v_1 as shown in Figure 2.

As shown in [10], any real $m \times n$ matrix A can be decomposed into $A = U\Sigma V^T$, where $U = [u_1, u_2, \ldots, u_m] \in R^{m \times m}$ and $V = [v_1, v_2, \ldots, v_n] \in R^{n \times n}$ are two orthogonal matrices, and Σ is a diagonal matrix with diagonal entries being the singular values of A: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(m,n)} \geq 0$. Column vectors u_i and v_i are unit vectors and are the i^{th} left and right singular vectors of A, respectively.

Along the direction of the first right singular vector, the projections of row vectors in A have the largest 2-norm, and along the second right singular vector direction, the projection 2-norm is the second largest, and so on. The singular values reflect the Euclidean lengths or 2-norms of the projections along the corresponding right singular vectors [10].

Although the left singular vectors of similar motions with different lengths are of different dimensions, the right singular vectors are of the same dimension. The singular values of matrix A are unique, and the singular vectors associated with distinct singular values are uniquely determined up to the sign, or a singular vector can have opposite signs [11]. For convenience, we will refer to the right singular vectors as singular vectors.

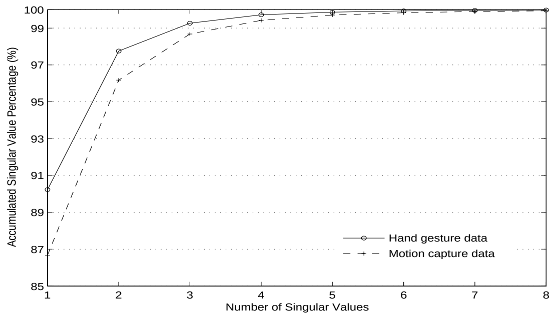

Figure 3. Accumulated singular value percentages in singular value sums for hand gesture data and human motion data. There are 22 singular values for hand gesture data and 54 singular values for motion capture data. The first singular values are more than 85% of the corresponding singular value sums.

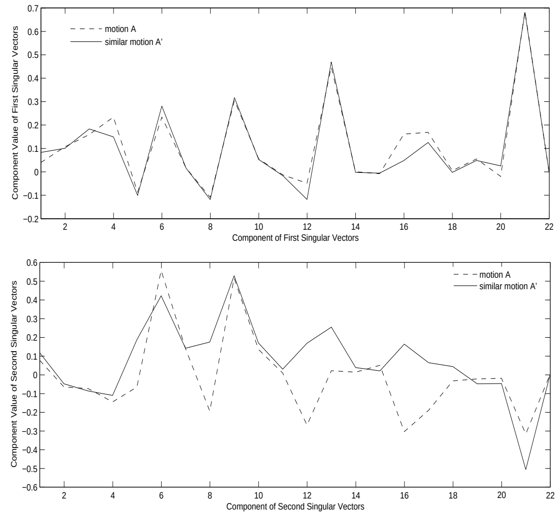

Figure 4. Singular vectors of similar motions. The first singular vectors are similar to each other, while other singular vectors, such as the second vectors as shown at the bottom, can be quite different.

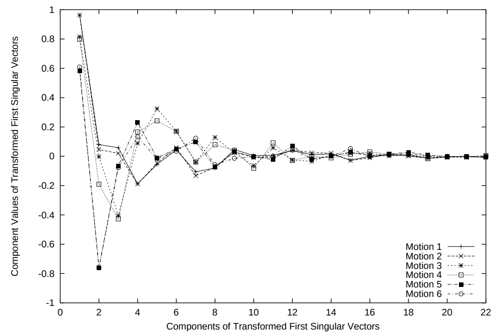

Figure 5. Component distributions of the transformed first singular vectors

IV. Feature Vector Extraction for Indexing

The trajectories of two motions in the high dimensional space should be similar to each other if the motions are similar. In other words, motion data matrices should have similar geometric structures if the corresponding motions are similar. Since the geometric similarity of matrix data can be captured by SVD, we propose to exploit SVD to generate representative vectors or feature vectors for motion matrices, and use these feature vectors to index the corresponding multi-attribute motion data.

As Figure 3 shows, the first singular values are the dominating ones among all singular values. Since the singular values reflect the magnitudes of the row vector projections along their corresponding singular vectors [10], we can say that the first singular vectors are the dominating vectors. If two motions are similar, their corresponding first singular vectors u_1 and v_1 should be mostly parallel to each other geometrically, so that $|u_1 \cdot v_1|$ = $|u_1||v_1||cos(\theta)| \doteq |u_1||v_1|$ = 1, where θ is the angle between the two right singular vectors u_1 and v_1, and $|u_1| = |v_1|$ = 1 by the definition of SVD. Similarly, the first singular vectors are also very likely to be different from each other when two motions are different. Other corresponding singular vectors may not be close to each other even if two motions are similar as shown in Figure 4. This suggests that the first right singular vectors can be used to prune the majority of different motions.

Since the dimensions of the first singular vectors of multi-attribute motion data are usually larger than 15, dimensionality reduction needs to be performed on them in order to avoid the so-called "curse of dimensionality." We use SVD further to reduce the dimensionality of the first singular vectors to be indexed. Let A be the matrix composing the first singular vectors of the motions to be indexed, and

$$A = W\Sigma Z^T$$

then $AZ = W\Sigma$ gives the projections of the first singular vectors onto the coordinate system spanned by the column vectors of Z [12], and for a singular vector u_1 of a query motion, $u_1 Z$ gives a corresponding projection of u_1 onto the system spanned by the column vectors of Z. For simplicity, let the projections of a first singular vector onto Z be a transformed first singular vector.

Due to singular value decomposition, the component variances of the transformed first singular vectors are the largest along direction z_1, and decrease along directions $z_2, \ldots, z_n$ as shown in Figure 5. The differences among the first singular vectors are optimally reflected in the first few dimensions of the transformed first singular vectors, hence we can index the first singular vectors by indexing only the first several components of the transformed singular vectors. Differences among all the other corresponding components are small even if motions are different, hence the other components can be truncated and the dimensionalities are reduced to the first few ones.

We refer to the transformed singular vectors after dimensionality reduction as the *feature vectors* of the motions. If the first component of a feature vector is negative, all components of this vector are negated to obtain a consistent sign for feature vectors of similar motions [4].

The feature vectors extracted above take only the most dominating first singular vectors. Since the second singular values can be much larger than others except the first singular values, we can take into account the associated second singular vectors. This can be done by concatenating the weighted second singular vectors $w_2 u_2$ to the weighted first singular vectors $w_1 u_1$ to generate composite vectors $w_1 u_1 w_2 u_2$, where $w_i = \sigma_i / \sum_{k=1}^{n} \sigma_k$. Similarly, the dimensionality of the concatenated vectors can be reduced as above to generate new feature vectors.

Each component of the feature vectors extracted as above is between -1 and 1, and the feature vectors can be used to prune irrelevant motions. It is worth noting that more singular vectors and singular values should be considered to recognize similar motions, as shown in [13].

V. INDEX TREE CONSTRUCTION

It can be observed that, as shown in Figure 5, similar motions have certain value ranges/intervals at each dimension of the feature vectors as extracted above. If the differences between the corresponding components of the feature vectors are too large, the associated motions cannot be similar. Hence each feature vector component can be used for pruning irrelevant motions, and an interval-based index tree can be constructed to index the feature vectors.

Let r be the dimension of the feature vectors, $r < n$. We designate one level of the index tree to each of the r dimensions. Let level 1 be the root node, level i includes nodes for dimension i, $i = 1, 2, \ldots, r$, and level r + 1 contains leaf nodes. Leaf nodes contain motion identifiers P_k, and non-leaf nodes contain entries of the form

$$(I_i, cp)$$

where I_i is a closed interval $[a, b]$ describing the component value ranges of the feature vectors at level i, $-1 \leq a, b < 1$. Each entry has the address of one child node, and cp is the address of the child node in the tree.

The width and boundary of interval I_i depend on the distribution of i^{th} component values of feature vectors and the possible variations of the i^{th} feature vector components of similar motions. Let δ_i be the maximum difference of the i^{th} feature vector components of any similar motions, let x_i and y_i be the respective minimum and maximum values of the i^{th} components of all feature vectors. We let the interval width be proportional to δ_i and use a parameter ϵ to adjust entry intervals. That is, the width of entry intervals at the i^{th} level is $\epsilon\delta_i$, and the number of entries of a node at level i is $\lceil (y_i - x_i)/(\epsilon\delta_i) \rceil$, limited by maximum number of entries per node allowed. We call parameter ϵ the entry interval factor hereafter.

A. Insertion and Searching

Real motions have different variations, causing variations in feature vector components. Hence, feature vectors of similar motions can be in different entries of a node. To speed up queries, we can insert feature vector IDs of similar motions in all possible entries of a node. For example, if feature vector components of two similar motions fall in entries e_1 and e_2, respectively at certain level, we insert both motion IDs in both e_1 and e_2. Hence searching of similar motions does not need visits of two entries.

Let the root node of a tree be T. The unique ID of a feature vector is inserted into a tree by comparing the i^{th} component c_i of the feature vector and the entry interval $[a, b]$ of the node traversed and can be inserted into multiple neighboring intervals:

- **Subtree Insertion:** If T is a non-leaf node, find all entries whose I_i's overlap with $[c_i - \epsilon\delta_i, c_i + \epsilon\delta_i]$. For each overlapping entry, find the subtree whose root node T is pointed to by cp of the overlapping entry.
- **Leaf Node Insertion:** If T is a leaf node, insert the motion pattern identifier P_k of the feature vector in T.

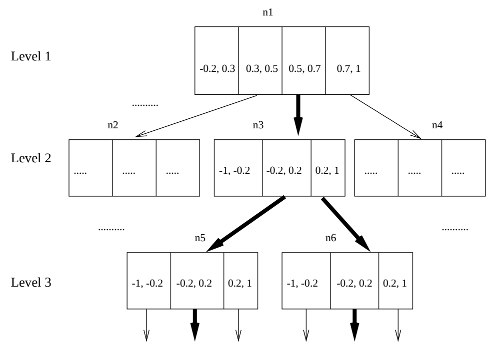

Figure 6. An index tree example showing three non-leaf levels. Bold lines show where a feature vector is to be inserted.

Figure 6 illustrates how to insert an example feature vector into the first three levels of an example index tree. Root node at level 1 has four entries, each of which has a child node at level 2. Each node at level 2 and level 3 has three entries, and each of which has a child node at one lower level. Given a feature vector f = (0.65, 0.15, -0.1, ...), and let δ_1 = 0.04, δ_i = 0.08 for $i \geq 2$, and ϵ = 1.0. Entries at the root node are checked with $[0.65 - 0.04, 0.65 + 0.04] = [0.61, 0.69]$. Only the third entry overlaps with it, hence the vector f is forwarded only to node n_3 of level 2. At level 2, the feature vector covering range is [0.15-0.08, 0.15+0.08] or [0.07, 0.23]. The second and third entries of node n_3 overlap with the feature vector covering range [0.07, 0.23], hence the feature vector will be forwarded to node n_5 and to node n_6 at level 3. At level 3, the feature vector covering range is [-0.1-0.08, -0.1+0.08] or [-0.16, -0.02]. Only the second entries of nodes n_5 and n_6 overlap with this range, so the nodes pointed by the second entries of nodes n_5 and n_6 will be traversed for insertion. This process goes on until the leaf nodes are traversed for holding P_k of the feature vector f.

A query searching can be very simple: find the entry of the node whose interval $[a, b]$ covers the i^{th} component c_i of the query feature vector and traverse to the corresponding child node pointed to by the entry. When a leaf node is reached, all the motion identifiers included in that leaf node are returned for the query. Since a node entry contains all possible similar motions in neighboring entries of the same node, only one entry is needed to be traversed for a search at each level of the tree, rather than multiple entries to be traversed as in [4].

B. Similarity Computation

After the index tree has been searched for a query, the majority of irrelevant motions should have been pruned, and similar motions and a small number of irrelevant motions are returned as the result of the query. To find out the motion most similar to the query, a similarity measure shown below as defined in [13] can be used to compute the similarity of the query and all the returned motions, and the motion with the highest similarity is the one most similar to the query.

$$\Psi(Q,P) = \frac{1}{2}\sum_{i=1}^{k}((\sigma_i/\sum_{i=1}^{n}\sigma_i + \lambda_i/\sum_{i=1}^{n}\lambda_i)|u_i \cdot v_i|)$$

where σ_i and λ_i are the i^{th} singular values corresponding to the i^{th} right singular vectors u_i and v_i of square matrices Q^TQ and P^TP of Q and P, respectively, and $1 < k < n$. Integer k determines how many singular vectors are considered and depends on the number of attributes n of motion matrices. Experiments with hand gesture motions (n = 22) and human body motions (n = 54) show that k = 6 is large enough without loss of pattern recognition accuracy in streams.

VI. Performance Evaluation

This section evaluates the performances of the proposed indexing structure, including the pruning efficiency, recall and query time for different tree configurations and different feature vectors. The pruning efficiency $\mathcal{P}$ is defined as follows.

$$\mathcal{P} = \frac{N_{pr}}{N_{ir}} \times 100\%$$

where N_{pr} is the number of irrelevant motions pruned for a query by the index tree, and N_{ir} is the total number of irrelevant motions in a database.

A. Motion Data Generation

Motion data was generated for hand gestures by using a data glove called CyberGlove and for dances and other human motions captured by using a motion capture system with 16 Vicon digital optical cameras. Motions are recorded at about 120 frames per second. Each motion frame is recorded by a high-dimensional vector, and continuous motion frames generate a multi-attribute data matrix for each isolated motion. For hand gestures, each of 22 sensors records the angular values of one joint of a hand. For motion capture (MoCap) data, eighteen segments jointly record the full body motions, and 3D coordinates for each segment joint are recorded, generating data matrices of 54 attributes.

The captured motion data had been transformed so that data matrices are the 3D positional coordinates of different segment joints relative to the pelvis joint. After this data transformation, similar motions performed at different locations, following different paths, or at different orientations have "similar" data matrices.

The proposed indexing approach works not only for 3D full body motions, but also for sub-body motions. When we are interested in motions of only arms or legs, indexing structures can be constructed for arms or legs only. Since motions of arms can have more variations than leg motions, we experimented with arm motions. Arm motions are the motions of arms when full body motions were performed, and were not generated separately.

One hundred and ten different hand gestures were generated, and each one was repeated for 3 times, resulting in 330 data matrices of 22 columns. Sixty two different full body motions, including Taiqi and dances were performed, and each one was repeated for 5 times, resulting in 310 MoCap data matrices of 54 columns for full body motions. Data matrices for motions of two arms have 24 columns. That is, there are 310 data matrices of 24 columns for 62 different arm motions.

B. Index Tree Configurations

We experimented with different tree configurations for hand gesture data and MoCap data. For hand gesture data of 22 attributes, feature vectors have 5 to 10 components, or trees of 5 to 10 levels were tested. For MoCap data of 54 attributes, trees of 5 to 12 levels were tested. The entry interval factors ϵ we tested range from 0.5 to 1.2. When ϵ = 1.0, the width of entry intervals at each tree level is the maximum difference of corresponding feature vector components of similar motions, determined by experiments on real motion data. The smaller the entry interval factors, the smaller the widths of entry intervals, and the more the number of entries in a node at all levels.

C. Pruning Efficiency

We issued one query for every one of the 330 hand gestures, 310 full body MoCap motions, and 310 arm motions. For each query of a hand gesture, there are two other similar gestures and 327 distinct gestures in the hand gesture database. For each query of full body motion and arm motion, there are 4 other similar motions and 305 distinct motions in the MoCap database, respectively.

Since the entry interval factor ϵ determines the widths of entry intervals and the numbers of motions an entry can cover, the larger the ϵ, the more irrelevant motions will be retrieved as shown in Figure 7 and Figure 8. When ϵ is fixed, trees with higher levels have higher pruning efficiency due to more feature components considered. Yet when the tree levels are more than 7, the increase in pruning efficiency is very small as shown in Figures 7 and 8.

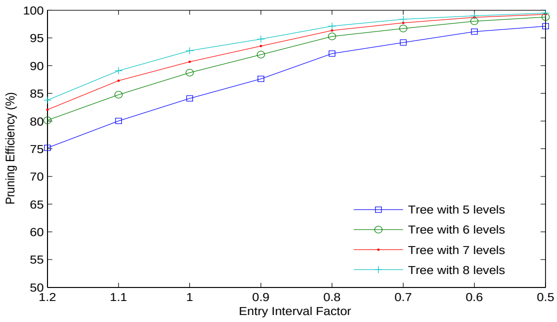

Figure 7. Pruning efficiency of trees with different levels and different entry interval factors for hand gesture data. Feature vectors were generated from two weighted singular vectors.

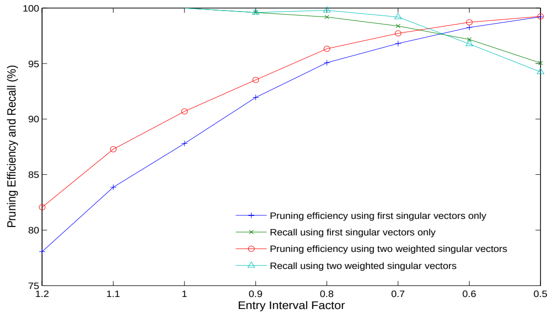

Figure 11. Pruning efficiency and recall of trees with different entry interval factors for two different feature vectors of hand gesture data. Tree level is seven.

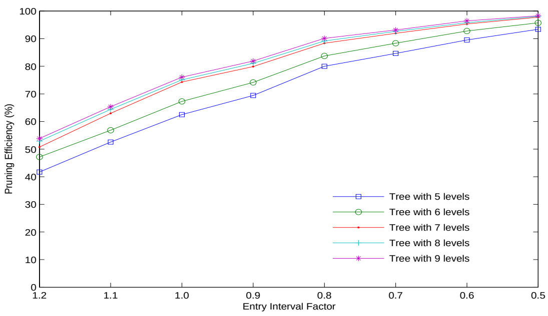

Figure 8. Pruning efficiency of trees with different levels and different entry interval factors for arms motion capture data. Feature vectors were generated from the first singular vectors.

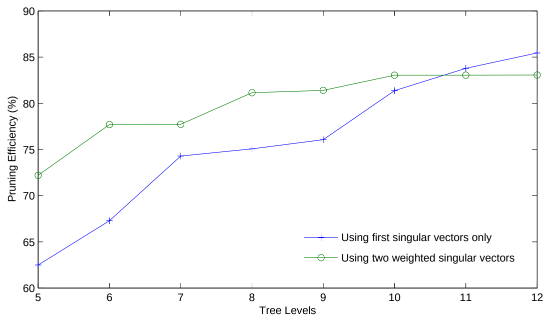

Figure 12. Pruning efficiency of trees with different levels for two different feature vectors of arms motion capture data. Entry interval factor ϵ = 1.0

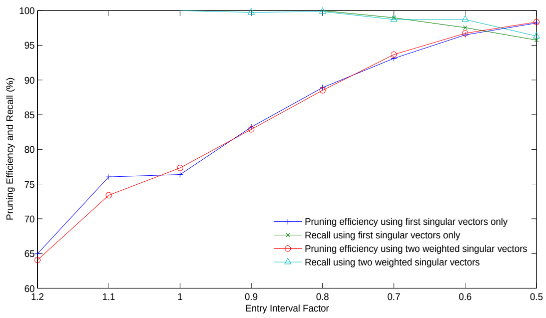

Figure 9. Pruning efficiency and recall of trees with different entry interval factors for two different feature vectors of full body motion capture data. Tree level is seven.

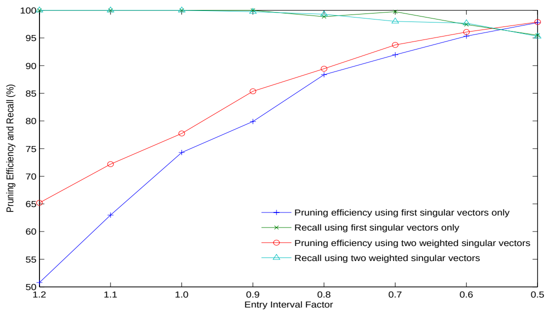

Figure 10. Pruning efficiency and recall of trees with different entry interval factors for two different feature vectors of arms motion capture data. Tree level is seven.

When the entry interval factor ϵ is no less than 1.0, all similar motions can be retrieved as shown in Figures 9-11 for trees of level 7. When ϵ is around 0.8, almost all similar motions can still be retrieved as indicated by the high recalls for different tree configurations. When ϵ is as small as 0.5, the most similar motions can still be retrieved and only a small number of less similar motions can be pruned. It can be observed that for trees with different entry interval factors ϵ, using feature vectors generated from two weighted singular vectors outperforms using feature vectors generated from only the first singular vectors in terms of high pruning efficiency and high recalls. Notice that when $\epsilon = 0.8$, the recalls are almost 100%, and the pruning efficiency is around 90% for arms and full body motions, and is around 95% for hand gestures.

Figures 12-14 compare the pruning efficiency for two different feature vectors. For hand gestures and arm motions, using feature vectors generated from two weighted singular vectors outperforms using feature vectors generated from only the first singular vectors when tree levels are no more than 10. When the full body motions are concerned, two weighted singular vectors outperform the first singular vectors when tree levels are no more than 8. The reason is that when tree levels or dimensionalities of feature vectors are small, more information is truncated. Hence, using two weighted singular vectors help reduce information loss due to feature vector truncation.

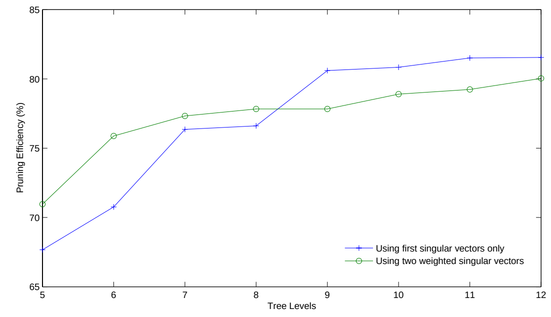

Figure 13. Pruning efficiency of trees with different levels for two different feature vectors of full body motion capture data. Entry interval factor ϵ = 1.0

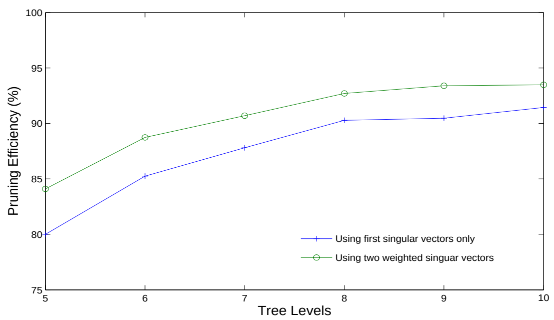

Figure 14. Pruning efficiency of trees with different levels for two different feature vectors of hand gesture data. Entry interval factor ϵ = 1.0

D. Computational Efficiency

We tested the average CPU time taken by a query using different tree configurations. All experiments are performed on one 3.0 GHz Intel processor of a GenuineIntel Linux box.

The search time of a query by using the proposed index structure takes less than 3 μsec as shown in Figure 15. As a comparison, the search time of a query by the algorithm in [4] can take several milliseconds as shown in Figure 16. As a tradeoff, the proposed approach in this paper takes a little longer for inserting feature vectors. Nevertheless, each insertion still takes less than 35 milliseconds as shown in Figure 17 and can be done off-line.

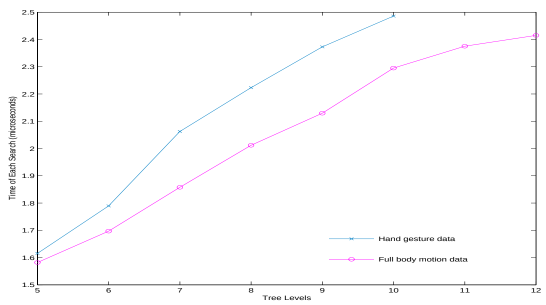

Figure 15. Search time for one query by the proposed indexing approach

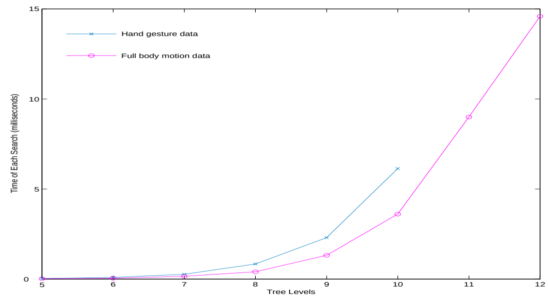

Figure 16. Search time for one query by the index structure as proposed in [4].

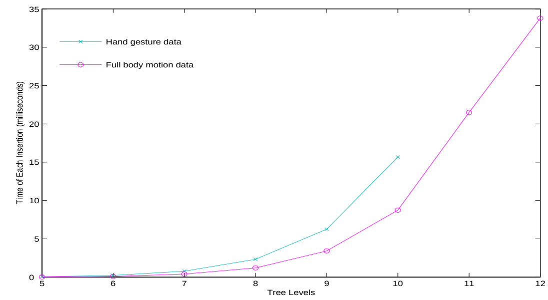

Figure 17. Time taken for inserting a new motion ID in the indexing tree

E. Performance Comparison

The MUSE indexing structure proposed in [6] works well on indexing synthetic hand gestures. It computes a lower bound to prune irrelevant motions at each indexing level. To compare its performance with that of our proposed indexing structure, we experimented with the real hand gestures and 3D motion capture data as shown in Figure 18. Experiments show that the lower bound defined in [6] is not tight enough for the real datasets, and our proposed approach gives high pruning efficiency for queries of real motions when all similar motions are to be retrieved. It takes MUSE more than 20 $msec$ for one query on hand gestures and arm motions, and 130 $msec$ for one query on full body motions. In comparison, it

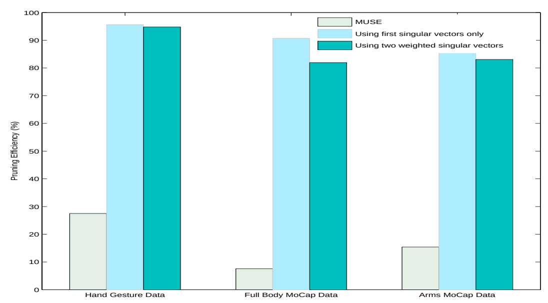

Figure 18. Maximum pruning efficiencies when all similar motions are to be retrieved

takes our proposed indexing structure less than 3 μsec for one query on the three datasets.

VII. CONCLUSIONS

This paper has proposed a novel approach for indexing real multi-attribute motion data of different lengths. Two different feature vectors are extracted from motion data matrices by using SVD properties, and an interval-based tree structure is proposed for indexing the feature vectors. Feature vector IDs can be inserted into multiple neighboring node entries to handle motion variations and can be in multiple leaf nodes. As an advantage of this design, search of similar motions can be done in only a few microseconds by traversing only one node once at each tree level, and up to 95 % different hand gestures and 91% captured human motions can be pruned.

The proposed indexing approach is applicable to both full body motions and sub-body such as arm motions, as well as hand gestures. The approach is based on the geometric structures as revealed by SVD of the motion data matrices. As long as the attributes of data sequences are correlated and have certain geometric structures such that the variances along the directions of the first right singular vectors are much larger than those along other directions, the proposed index structure should be applicable to the sequences. Hence, we believe it can be applied to other multi-attribute data of different lengths.

ACKNOWLEDGMENT

We would like to thank Gaurav Pradhan and Punit R Kulkarni for helping generate the motion data used for this work.

REFERENCES

[1] T. Kahveci, A. Singh, and A. Gurel, "Similarity searching for multi-attribute sequences," in *Proceedings. of 14th Int'l Conference on Scientific and Statistical Database Management*, July 2002, pp. 175 – 184.
[2] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung, "Similarity search for multidimensional data sequences," in *Proceedings. of 16th Int'l Conference on Data Engineering*, Feb./Mar. 2000, pp. 599 – 608.
[3] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *SIGMOD*, August 2003, pp. 216–225.
[4] C. Li, G. Pradhan, S. Zheng, and B. Prabhakaran, "Indexing of variable length multi-attribute motion data," in *Proceedings of the Second ACM International Workshop on Multimedia Databases 2004*, November 2004.
[5] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle, "Indexing large human-motion databases," in *Proceedings of 30th VLDB Conference*, September 2004.
[6] K. Yang and C. Shahabi, "Multilevel distance-based index structure for multivariate time series," in *Proceedings of 12th International Symposium on Temporal Representation and Reasoning (TIME 2005)*, June 2005.
[7] M. Muller, T. Roder, and M. Clausen, "Efficient content-based retrieval of motion capture data," *ACM Transactions on Graphics (TOG)*, vol. 24, pp. 677–685, 2005.
[8] F. Liu, Y. Zhang, F. Wu, and Y. Pan, "3d motion retrieval with motion index tree," *Computer Vision and Image Understanding*, vol. 92, pp. 265–284, 2003.
[9] G. Liu, J. Zhang, W. Wang, and L. McMillan, "A system for analyzing and indexing human-motion databases," in *SIGMOD*, June 2005.
[10] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, Maryland: The Johns Hopkins University Press, 1996.
[11] B. D. Schutter and B. D. Moor, "The singular value decomposition in the extended max algebra," *Linear Algebra and Its Applications*, vol. 250, pp. 143–176, 1997.
[12] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently supporting ad hoc queries in large datasets of time sequences," in *SIGMOD*, May 1997, pp. 289–300.
[13] C. Li and B. Prabhakaran, "A similarity measure for motion stream segmentation and recognition," in *Proceedings of the Sixth International Workshop on Multimedia Data Mining*, August 2005.

Chuanjun Li is currently a Ph.D. candidate in Computer Science at the University of Texas at Dallas. His Ph.D. research focuses primarily on efficient segmentation and recognition of 3D human motion streams, and development of indexing and clustering techniques for the multi-attribute motion data as well as classification of motion data.

B. Prabhakaran is currently with the Department of Computer Science, University of Texas at Dallas. Dr B. Prabhakaran has been working in the area of multimedia systems: multimedia databases, authoring & presentation, resource management, and scalable web-based multimedia presentation servers. He has published several research papers in prestigious conferences and journals in this area.

Dr. Prabhakaran received the NSF CAREER Award FY 2003 for his proposal on Animation Databases. Dr. Prabhakaran has served as an Associate Chair of the ACM Multimedia'2003 (November 2003, California), ACM MM 2000 (November 2000, Los Angeles), and ACM Multimedia'99 conference (Florida, November 1999). He has served as guest-editor (special issue on Multimedia Authoring and Presentation) for ACM Multimedia Systems journal. He is also serving on the editorial board of Multimedia Tools and Applications journal, Kluwer Academic Publishers. He has also served as program committee member on several multimedia conferences and workshops. Dr. Prabhakaran has presented tutorials in several conferences on topics such as network resource management, adaptive multimedia presentations, and scalable multimedia servers.

B. Prabhakaran has served as a visiting research faculty with the Department of Computer Science, University of Maryland, College Park. He also served as a faculty in the Department of Computer Science, National University of Singapore as well as in the Indian Institute of Technology, Madras, India.