A HYBRID APPROACH FOR HUMAN MOTION RETRIEVAL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Yağız Salor
October, 2015

A HYBRID APPROACH FOR HUMAN MOTION RETRIEVAL
By Yağız Salor
October, 2015

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Bülent Özgüç (Advisor)

Assoc. Prof. Dr. Tolga Çapın (Co-Advisor)

Prof. Dr. Uğur Güdükbay

Prof. Dr. Veysi İşler

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

A HYBRID APPROACH FOR HUMAN MOTION RETRIEVAL

Yağız Salor
M.S. in Computer Engineering
Advisor: Prof. Dr. Bülent Özgüç
Co-Advisor: Assoc. Prof. Dr. Tolga Çapın
October, 2015

Retrieving similar motions from motion databases has become an essential topic of computer animation research. The use of binary geometric features and inverted indexes is one of the efficient solutions to this problem. This approach can be used with variation and inexactness algorithms for fuzzy searches. However, close similarity searches are not possible. In addition, the process is not automatic since it needs user input for selecting binary features to use. In another efficient approach, k-d tree with medium sized numerical based feature sets and shortest path search on a directed graph are used. However, it does not propose any tool for fuzzy searches. In this thesis, we propose a hybrid approach that utilizes numerical based feature sets, k-d tree based indexing structure and inverted index based motion matching technique. Our hybrid approach does not need user input, can be used in environments requiring close similarity of motions and can be used with variation and inexactness algorithms. Our results show that our hybrid approach is useful and efficient for similarity searches on motion databases.

Keywords: character animation; human motion retrieval; motion capture; motion retrieval.

ÖZET

İNSAN HAREKETİ ERİŞİMİ İÇİN MELEZ BİR YAKLAŞIM

Yağız Salor
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Prof. Dr. Bülent Özgüç
Eş-Tez Danışmanı: Doç. Dr. Tolga Çapın
Ekim, 2015

Hareket veritabanları üzerinden benzer hareket erişimi bilgisayar animasyonu araştırmalarının önemli bir konusu haline gelmiştir. İkili değerli geometrik özellik ve ters indis kullanımı bu problemin verimli çözümlerinden birisidir. Bu yaklaşım bulanık aramalar için eşitlilik ve hatalılık algoritmaları ile birlikte kullanılabilmektedir. Fakat yakın benzerlik aramaları bu yaklaşım ile mümkün değildir. Buna ek olarak bu yaklaşım ikili değerli geometrik özellik seçiminde insan girişi gerektirdiğinden otomatik değildir. Başka bir verimli yaklaşımda, orta boyutta rakamsal özellik kümeleri, k-d ağacı ve yönlü çizge üzerinde en kısa yol arama kullanılmıştır. Fakat bu yaklaşım bulanık aramalar için bir araç sunmamaktadır. Bu tezde, rakamsal özellik kümeleri, k-d ağacı tabanlı dizinleme yapısı ve ters indisler kullanan bir melez yaklaşım öneriyoruz. Bizim melez yaklaşımımız kullanıcı girişine ihtiyaç duymamaktadır ve çeşitlilik algoritmaları ile birlikte kullanılabilmektedir. Sonuçlarımız melez yaklaşımımızın hareket veritabanı üzerinde benzerlik aramaları için kullanılabilir ve verimli olduğunu göstermektedir.

Anahtar sözcükler: karakter animasyonu; insan hareketi erişimi; hareket yakalama; hareket erişimi .

Acknowledgement

I am grateful to my advisors Prof. Dr. Bülent Özgüç and Assoc. Prof. Dr. Tolga Çapın for their support, suggestions and encouragement during my study.

I would like to thank my jury members Prof. Dr. Uğur Güdükbay and Prof. Dr. Veysi İşler for reviewing my thesis.

I would also like to thank my coworkers at TÜBİTAK-BİLGEM-YTE for their efforts to motivate me.

Lastly, I would like give special thanks to my mother and father for their love and continuous support during my education.

The data used in this thesis was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Different kinds of realistic human motion data is needed by video games and animation movies. However, it is extremely difficult for animation artists to manually create realistic human movements. For that reason, the usage of previously recorded human motion capture data is inevitable. Free to use motion databases such as CMU [1] and HDM05 [2] that provide wide selection of motion types are available on the internet. These databases contain hours of motion capture data that are classified into tens of motion classes. Therefore, retrieving similar motions on these databases became an essential topic of computer animation research.

Many computer animation researchers have developed numerous kinds of techniques to bring a solution to this problem. The approaches can be grouped into two by their similarity notion as *numerical* and *logical*. Müller et al. [3] propose an efficient approach that uses logical similarity notion called boolean feature vectors that can handle temporal and spatial variations between compared motions. Fuzzy search with adaptive segmentation provides more variation handling in both domains. In addition, their approach can be extended to include k-mismatch [4] concept that brings inexactness to the retrieval process. However, in this approach the user needs to select relevant boolean features for each type of query to get high quality results.

Krüger et al. [5] point out that boolean features are not suitable in environments requiring close similarity of motions. Instead, they use small and medium feature sets with numerical similarity notion. Their approach does not require any kind of user input i.e. it is automatic. However, their approach does not have any mechanism that brings inexactness to the queries. Since their motion matching algorithm is completely different from the algorithm of Müller et al., it is not possible utilize the algorithms of latter into the former one.

In this thesis, we propose a hybrid approach that utilizes numerical based feature sets and k-d tree [6] based indexing structure of Krüger et al. and inverted index based motion matching technique of Müller et al. [3]. Our hybrid approach does not need user input, can be used in environments requiring close similarity of motions and can be used with variation and inexactness algorithms of Müller et al.

The organization of this thesis is as follows. Chapter 2 provides background information on human motion retrieval. In Chapter 3, our hybrid approach for human motion retrieval is explained. Chapter 4 provides results of our human retrieval system on a sample dataset compiled from CMU Mocap Database [1]. Finally, Chapter 5 concludes.

Chapter 2

Background

2.1 Motion Databases

The existence of large human motion capture databases such as CMU [1] and HDM05 [2] has led the computer scientists to study matching and retrieval of similar motions in these databases. These studies can be compared by features that are selected for similarity search, the indexing structure and the approach used by detecting the similarity.

2.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is used to find optimal alignment between two given sequences [7].

Human motion data contains set of poses that are comprised by set of joint angles between the bones in the human skeleton. Each human motion file has certain number of poses for each second in the animation that is defined by frame rate. Therefore, human motion retrieval is considered to be part of time series databases research. According to [8], there exist approaches that use Euclidean

distance for time series data. However, this similarity measure is very sensitive for the variations in time. Using the result of a simple experiment, they prove that DTW is more suitable similarity measure than Euclidean distance for time series data. Therefore, DTW is a widely used technique in human motion retrieval systems research.

Cardler et al. [9] introduce a content-based method for retrieving perceptually similar motions that provides editing for animators. Their method rely on Minimum Bounding Rectangles and R-trees for indexing the poses. Similar motion retrieval is done with the help of DTW and Longest Common Subsequence similarity metrics which enables temporal variations. The ability of selecting body areas that would be used in the matching process enables spatial variations.

Chiu et al. [10] propose another DTW based approach for content-based human motion retrieval. They suggest an index map structure for the poses in the human motion data. The human skeleton is divided into segments, then for each pose in the motion the sum of these segments are used as features. This idea helps to prevent curse of dimensionality since the poses in their data has high dimensions. In the matching process, start and end frames of the query motion are used in the index map structure to find similar candidate motions. Finally, similarity of these candidate motions with the query motion are evaluated using DTW.

Keogh et al. [11] point out the limitations of DTW. They explain that DTW is used for time alignment in the human retrieval systems but it can only provide local scaling i.e. only local variations in time is handled. They propose a novel approach based on bounding boxes which supports global (uniform) scaling and is faster than DTW. Uniform scaling perform better than DTW.

2.3 Logical vs. Numerical Similarity

Kovar et al. [12] advocate that logically similar motions need not be to be numerically similar. Logically similar motions could have different poses. For example, three punching motions are given in Figures 2.1, ..., 2.4. Even if all of those are punching motion, they have different numerical hand, arm, knee and feet positions. These differences are more noticeable in the second frames of these figures.

These kind of motions may not be retrieved by previously mentioned approaches. In order to prevent these misses, Kovar et al. introduce the technique named multi-step search. They use numerically similar results as new queries to find more similar results different than the previous step. Secondly, they compute average distance between similar poses and use them to compare with predefined certain value. This idea helps finding similar motions since they should have similar sequence of events. Lastly, they introduce a data structure called match web that increases speed of returning similar motions. However, match web structure has quadratic running time and has quadratic memory requirement to the number of poses in the database.

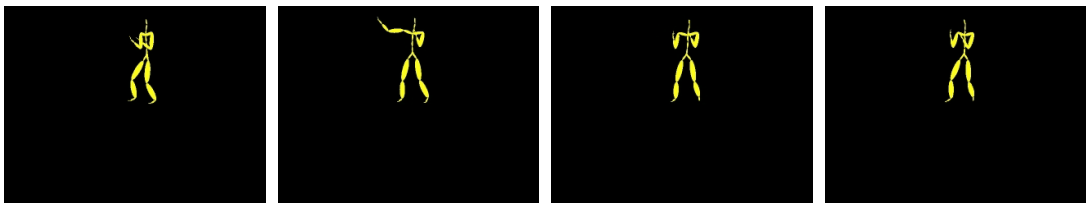


Figure 2.1: Punch motion 1

Figure 2.2: Punch motion 2

Figure 2.3: Punch motion 3

Figure 2.4: Punch motion 4

The shortcomings of numerical similarity is also pointed out by Müller et al. [3]. They define boolean feature sets for the human motion in order to eliminate the shortcomings of numerical similarity measures. These geometric boolean features returns a boolean results according to the positions of specified geometric features like positions of the points. Their approach handles both spatial and temporal variations in the comparisons of the motions. They use standard information retrieval techniques instead of DTW, which has a significant performance benefit. Building the index has a time complexity of $O(n)$, which is better than DTW based methods and match webs. However, this method needs user interaction. Appropriate features are need to be selected by the user for each kind motion. In addition, Krüger et al. [5] mention that this approach cannot handle close numerical similarity of motions. This is the main limitation of logical similarity measures.

2.4 Dimensionality Reduction

High dimensionality features of human motion data makes the retrieval process costly. In order to overcome this problem, dimensionality reduction techniques

have been used. Chai et al. [13] introduce a system that can reconstruct user motion from low dimensional control signals that is produced by videos recorded by two cameras. A graph of nearest neighbors is used for retrieving motion files that are similar to given control signals. The search is done by the use of temporal coherence of control signals. The algorithm chosen for dimensionality reduction affects the performance of the system significantly. The paper compares their own dimensionality reduction algorithm with existing ones such as Principal Component Analysis (PCA), Nonlinear Dimensionality Reduction and Local Linear Models (LLM).

Another approach to overcome curse of dimensionality of human motion data is to use smaller subsets of the human motion data that can produce acceptable results in practical use. Krüger et al. [5] demonstrate that the usage of small or medium dimensional feature sets of human motion data can be useful for pose matching in a similarity search. They perform nearest neighbor searches on different datasets that have dimensions ranging from 15 to 90 on a k-d tree [6] based indexing structure. They also introduce a novel graph based approach for motion matching that makes use of nearest neighbor searches on a k-d tree, which returns equivalent results to the approaches that utilizes DTW. However, their approach does not have a mechanism that provides inexactness for the search query such as k-mismatch [4].

Chapter 3

A Hybrid Approach for Human Motion Retrieval

We propose a hybrid approach for human motion retrieval. It can be used to retrieve similar motions from motion capture databases. The approach is based on kd-tree based indexing technique proposed by Krüger et al. [5] and inverted list based motion retrieval approach proposed by Müller et al. [3].

3.1 Motion Data

The motion data format used in this research is Acclaim ASF/AMC [14], which is developed by computer game company Acclaim. ASF/AMC data is comprised of two different file types, Acclaim Skeleton File (ASF) and Acclaim Motion Capture (AMC). ASF file holds fixed skeletal information of an actor whereas AMC file contains dynamic information of recorded motion. This design of the data format allows storing one ASF file for each actor and many AMC files for each recorded motion of that actor.

A base pose is defined in the ASF file as a starting point for the motion data.

The skeleton is defined as hierarchy of bones which are also called as segments. Figure 3.2 shows a sample skeletal hierarchy

The motion data is defined as a sample at a time in the AMC file. Each time sample contains position information of each bone segment in the corresponding ASF file. Figure 3.1 shows sample dancing motion that is sequence of skeletal poses of selected time samples.

Figure 3.1: Sample dancing motion that consists of sequence of skeletal poses

3.2 Notations

We use a notation that is influenced by the notations of Müller et al [3] and Krüger et al. [5].

J: a joint in the skeleton,

P: a pose which consists of joints,

M: a motion capture data stream which consist of poses i.e. each motion file in the database,

D: a motion capture database which consists of n motion capture data streams,

S: a similar pose,

Q: a motion capture data stream which used as a query,

R: a motion capture data stream which is resulted by the motion matching

progress,

F: the feature set that is used for inserting poses into the k-d tree and retrieving poses from the k-d tree,

n: total number of poses in the motion capture database D,

p: the number that defines how many number of poses P of each motion capture data stream M in the database D are going to be skipped while inserting them into the k-d tree indexing structure, and

k: the number of neighbors that are going to be retrieved from the database for each pose P in the query Q.

3.3 Feature Set

In this motion retrieval system we use ${F_E}^{15}$ feature set defined by Krüger et al. [5]. ${F_E}^{15}$ is used with numerical similarity measures and it consists of positions of 4 end-effectors and head. More specifically, these features are x, y and z positions of joints named "lhand", "rhand", "lfoot", "rfoot" and "head", which is shown in Figure 3.2. Therefore, ${F_E}^{15}$ is a 15 dimensional feature set which can be classified as medium.

Krüger et al give the motivation behind the ${F_E}^{15}$ with the following facts:

- "As is well known, the geometry of anthropomorphic limbs is fully determined by the positions of the end- effectors, their orientation, and one single additional scalar quantity so called swivel angle" [15].
- Orientations of the end effectors and swivel angles are dependent on the end effector positions.
- Similar body positions have similar end effector and head positions with little variations.

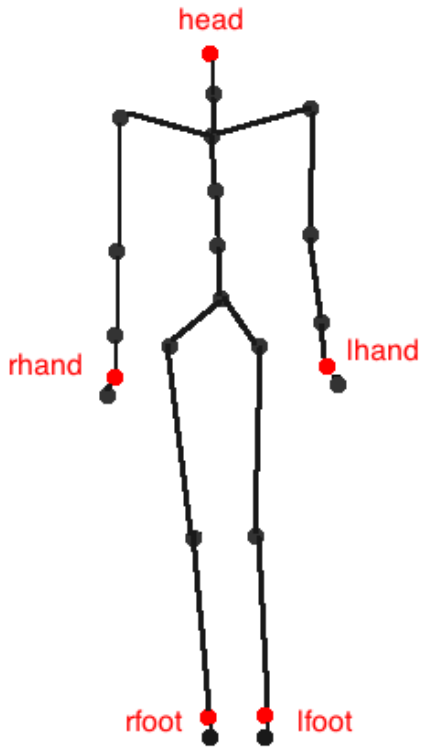


Figure 3.2: The skeleton structure that is used in our system. Each joint in the skeleton is shown with black dots. The joints that are included in the feature set are indicated with red dots.

They define more detailed pose based feature sets ${F_E}^{30}$ and ${F_E}^{39}$, which have 30 and 39 dimensions, respectively. They perform systematic comparisons that are based on pose level and motion segment level over different size of databases to validate efficiency of ${F_E}^{15}$. They show that these higher dimensional feature sets brings little or no advantage over the ${F_E}^{15}$, which is fastest in the nearest neighbor searches.

3.4 Pose Matching

In our system, pose matching is performed by the use of k nearest neighbor searches on a k-d tree [6] space partitioning data structure. A k-d tree is a multidimensional binary search tree. Building a k-d tree of n elements has $O(nlg(n))$ time

complexity. Performing a k nearest neighbor search on a k-d tree is $O(klg(n))$.

Figure 3.3 shows a k-d tree with two dimensions x and y with the points (4,5), (3,4), (6,7), (2,3), (3,5) inserted with the given order. The space decomposition of given k-d tree is shown in Figure 3.4.

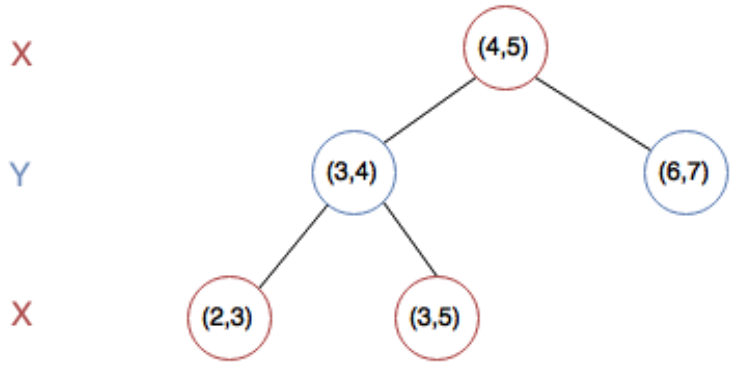


Figure 3.3: k-d tree for two dimensions x and y with the points (4,5), (3,4), (6,7), (2,3), (3,5) inserted in the given order

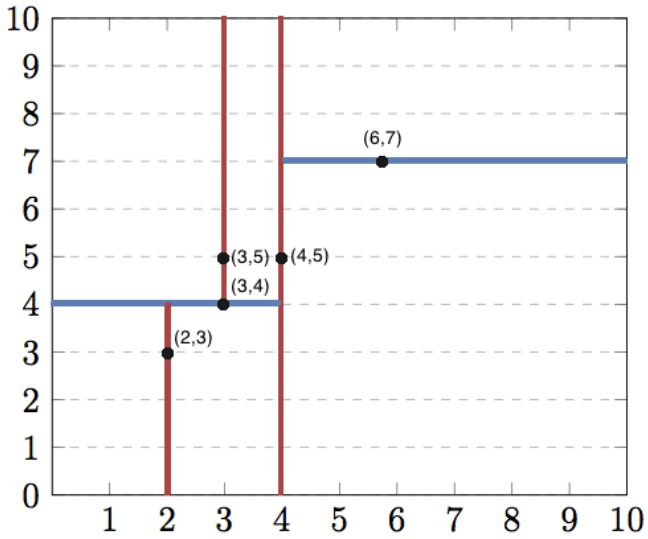


Figure 3.4: Space decomposition of the k-d tree with the points (4,5), (3,4), (6,7), (2,3), (3,5) inserted in the given order

In our hybrid approach, we use the geometric feature set $F_E{}^{15}$, which has 15

dimensions, as described in the Section 3.3. Therefore, we use a k-d tree of 15 dimensions.

3.5 Motion Retrieval

Our hybrid motion retrieval approach can be decomposed into three steps:

1. k-d tree construction,
2. pose matching with nearest neighbor search, and
3. motion matching.

The first two parts are based on the approach of Krüger et al. [5] and the last part is based on the approach of Müller et al. [3]

3.5.1 k-d tree Construction

The first step of our motion matching approach is k-d tree construction for a motion database D, which has n poses with respect to the our selected feature set ${F_E}^{15}$. It is actually a preprocessing step that can be performed once and can be used by as many searches as needed.

For each motion capture data stream $M = [P_1...P_f]$ in motion database D, we select one pose P_i for p number poses in the stream. This p is a pre-defined parameter and can be set to frame rate of the M or any number, e.g. just 1 for a greater accuracy.

The number of poses m to be inserted to the k-d tree is $m = n/p$. Therefore, the time complexity of this preprocessing step is $O(mlg(m))$.

3.5.2 Nearest Neighbor Search

$S(P_i)$ denotes the set of k nearest neighbors of one pose P. In the second step, we perform f/p number of k-NN searches to find set of neighbors $S(P_i)$ of each p pose in the query motion capture data stream $Q = [P_1...P_f]$, which has f number of poses. The time complexity of each retrieval is $O(klg(m))$.

3.5.3 Motion Matching

In the third step, we perform exact motion matching.

3.5.3.1 Exact Matching

This part is based on the exact hit approach of Müller et al. [3]. All of the motions in the motion database (D) are considered as one big motion file for simplicity.

This part utilizes inverted lists. Poses are used as index words. The time and space needed to build the indexing structure is $O(n)$. Two different inverted indexes $L_i(P)$ and $L_m(P)$ are needed.

$L_i(P)$: Returns the index of the given pose P in the database D.

$L_m(P)$: Returns the motion data stream M of a given pose P.

The set of starting indexes H_D of all similar matches of the given $Q = \{P_1, P_2, ..., P_L\}$ in the D is

$$H_D(Q) = \bigcap_{l \epsilon [1:L]} (L(S(P_l)) - l + 1). \tag{3.1}$$

when $L(S(P_l))$ is

$$L(S(P_l)) = \bigcup_{s \epsilon S(P_l)} (L(s)). \tag{3.2}$$

Here is an example, Let the database be

$$D = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7\} \tag{3.3}$$

which is the concatenation of motions

$$M_1 = \{P_1, P_2\}, \tag{3.4}$$

$$M_2 = \{P_3, P_4, P_5\}, \tag{3.5}$$

$$M_3 = \{P_6, P_7\}. \tag{3.6}$$

Let the query motion be

$$Q = \{P_8, P_9, P_{10}\}. \tag{3.7}$$

Let the nearest neighbors of each pose in the query after the similarity search on the k-d tree be

$$S(P_8) = \{P_3, P_6\}, \tag{3.8}$$

$$S(P_9) = \{P_4, P_7\}, \tag{3.9}$$

$$S(P_{10}) = \{P_5, P_7\}. \tag{3.10}$$

Then, index numbers of these similar poses are retrieved using inverted indexes

$$L_i(S(P_8)) = \{3, 6\}, \tag{3.11}$$

$$L_i(S(P_9)) = \{4, 7\}, \tag{3.12}$$

$$L_i(S(P_{10})) = \{5, 7\}. \tag{3.13}$$

After applying the formula on we get the following result

$$\begin{aligned} H_D(Q) &= (L_i(S(P_8)) - 1 + 1) \cap (L_i(S(P_9)) - 2 + 1) \cap (L_i(S(P_{10})) - 3 + 1) \\ &= \{3, 6\} \cap \{3, 6\} \cap \{3, 5\} \\ &= \{3\}. \end{aligned} \tag{3.14}$$

This means there is one match which has an index of 3 in the database. Then we use our other inverted index M to find which motion capture data stream it corresponds

$$L_m(P_2) = M_2. \tag{3.15}$$

Then we conclude that M_2 is the only matching motion for our query motion Q in our database D.

Chapter 4

Results

4.1 Pose Matching

4.1.1 Performance

Table 4.1 shows k-d tree building times with respect to the change in total number of poses in the database. The process has a time complexity of $O(nlg(n))$ where n is the total number of poses in the database. The plot of the building time is shown on Figure 4.1.

Total Number of Poses	k-d tree construction time (s)
1000	5.238
2000	15.211
3000	25.070
4000	36.655
5000	52.625
6000	70.506
7000	84.097
8000	95.309
9000	121.801
10000	144.312

Table 4.1: k-d tree construction time (s)

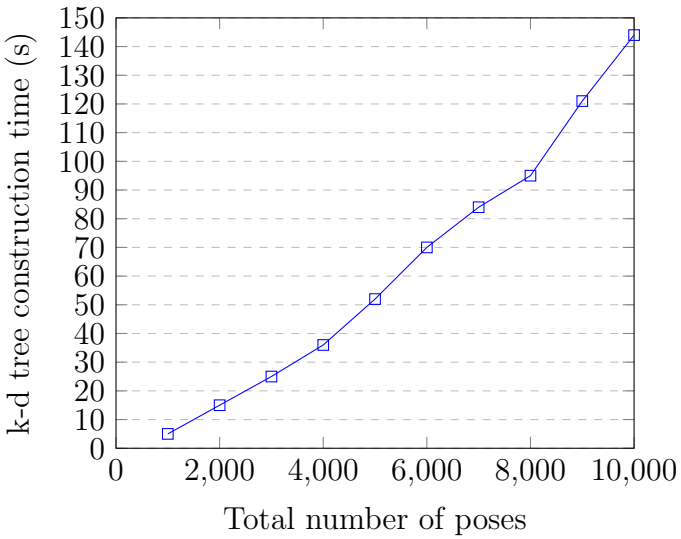


Figure 4.1: k-d tree construction time (s)/Total number of poses

Figure 4.4 shows an illustration of pose matching process that is performed by k nearest neighbor search on a k-d tree. The 50 neighbors of features “lhand”, “rhand”, “lfoot”,“rfoot” and “head” are shown as green dots. The nearest neighbor queries of each of these poses are performed very fast. Table 4.2 on page

shows these fetching times which are in miliseconds. Similar motion fetching has a time complexity of $O(klg(n))$ where n is the number of poses in the database. The plot of the pose fetching time with respect to the change in the total number of poses is shown in Figure 4.2. The plot of the pose fetching time with respect to the change in the k number in the nearest neighbor search is shown in Figure 4.3. These values are calculated by averaging 2000 k nearest neighbor searches in order to provide accuracy.

No of Poses (n) / k-NN (k)	k=50	k=100	k=150	k=200	k=250	k=300
n=1000	0.34	0.39	0.41	0.44	0.46	0.49
n=2000	0.71	0.81	0.89	0.97	1.03	1.05
n=3000	1.07	1.19	1.38	1.41	1.46	1.60
n=4000	1.45	1.68	1.80	1.97	2.08	2.18
n=5000	1.82	2.17	2.35	2.50	2.68	2.79
n=6000	2.22	2.79	2.92	3.04	3.33	3.44
n=7000	2.5	2.9	3.07	3.27	3.56	3.89
n=8000	3.01	3.66	3.72	3.88	4.13	4.61
n=9000	3.56	4.23	4.55	5.10	5.22	5.52
n=10000	4.82	5.1	5.39	5.43	6.03	6.72

Table 4.2: Similar pose fetching times (ms)

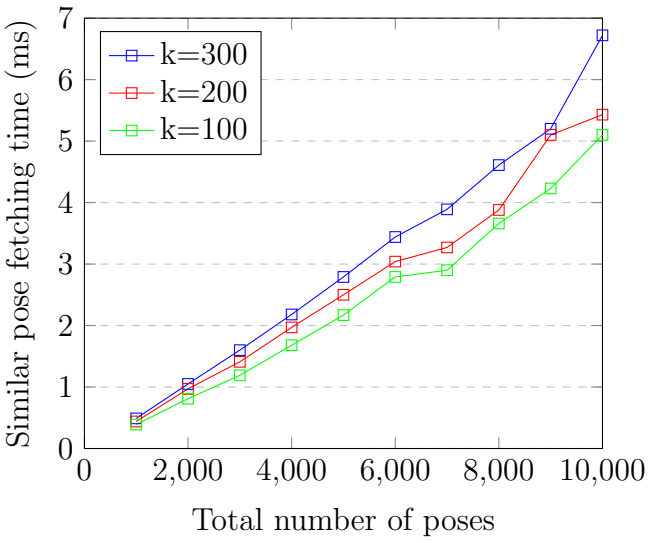


Figure 4.2: Similar pose fetching time (ms)/Total number of poses

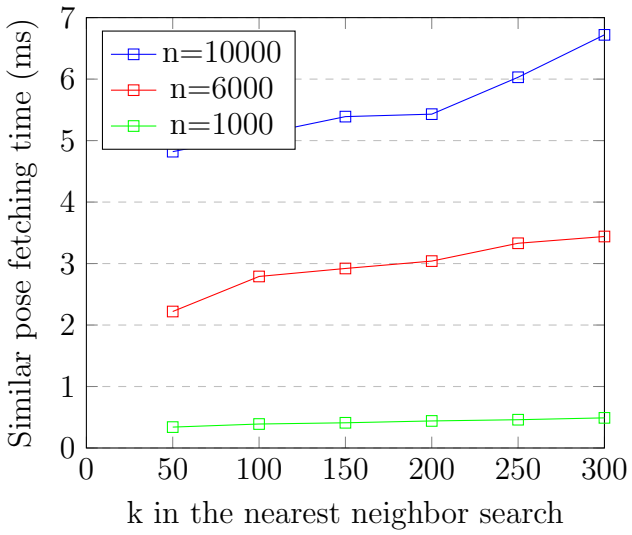
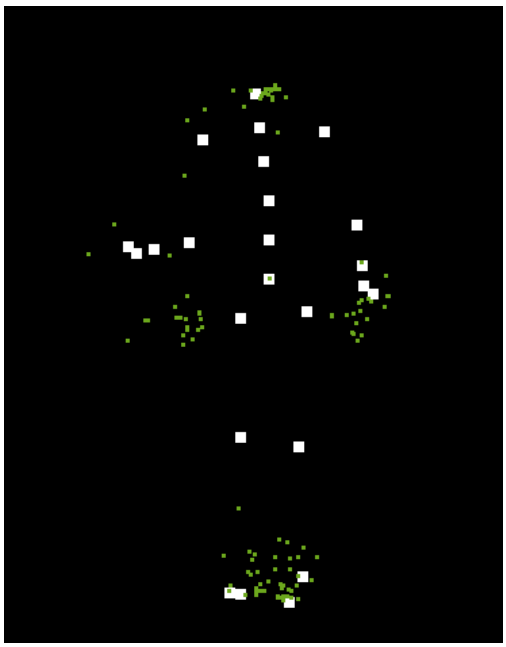
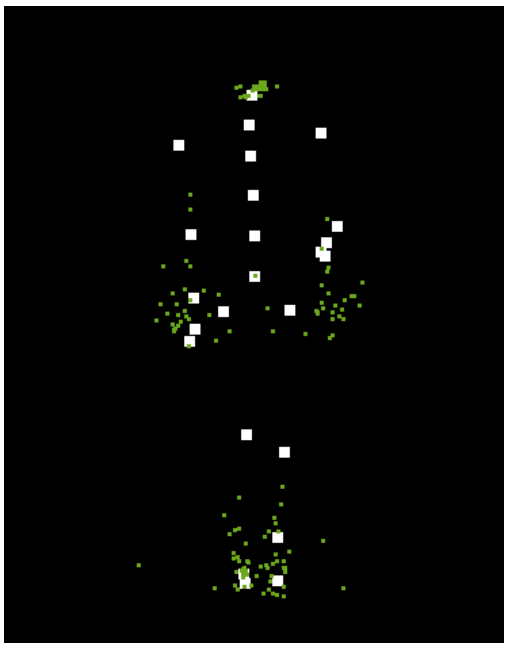


Figure 4.3: Similar pose fetching time (ms)/k in the nearest neighbor search

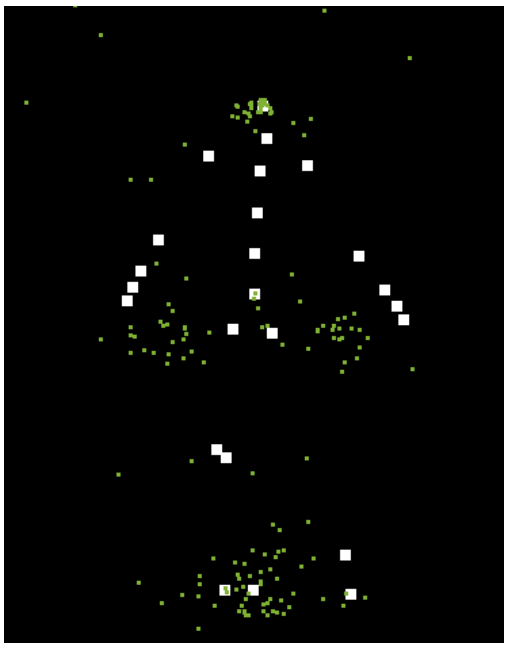
4.1.2 Visual Results



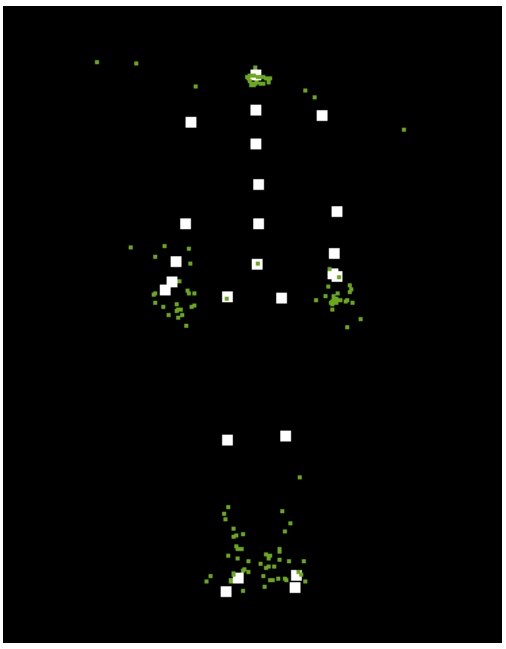
(a) A pose from ball dribbling motion



(b) A pose from a walking motion



(c) A pose from a dancing motion



(d) A pose from jumping motion

Figure 4.4: An illustration of pose matching process, which is performed by k nearest neighbor searches on a k-d tree. The 50 neighbors of features "lhand", "rhand", "lfoot", "rfoot" and "head" are shown as green dots.

4.2 Motion Matching

4.2.1 Performance

Table 4.3 shows motion matching times in miliseconds. The plot of the motion matching time with respect to the change in the total number of poses is shown in Figure 4.5. The plot of the motion matching time with respect to the change in the k number in the nearest neighbor search is shown in Figure 4.6. These values are calculated by averaging 2000 motion matching queries in order to provide accuracy.

No of Poses (n) / k-NN (k)	k=50	k=100	k=150	k=200	k=250	k=300
n=1000	2.94	3.21	3.47	3.75	3.91	4.11
n=2000	6.03	6.61	7.02	7.39	7.56	7.76
n=3000	8.01	9.10	9.95	10.44	10.53	10.95
n=4000	10.92	12.65	13.75	14.49	14.57	15.02
n=5000	12.48	15.24	16.57	17.93	18.10	18.73
n=6000	14.91	17.81	20.09	21.91	22.49	23.28
n=7000	17.44	20.73	22.65	24.49	25.37	26.30
n=8000	19.95	22.80	23.84	25.80	26.44	28.05

Table 4.3: Similar motion matching times (ms)

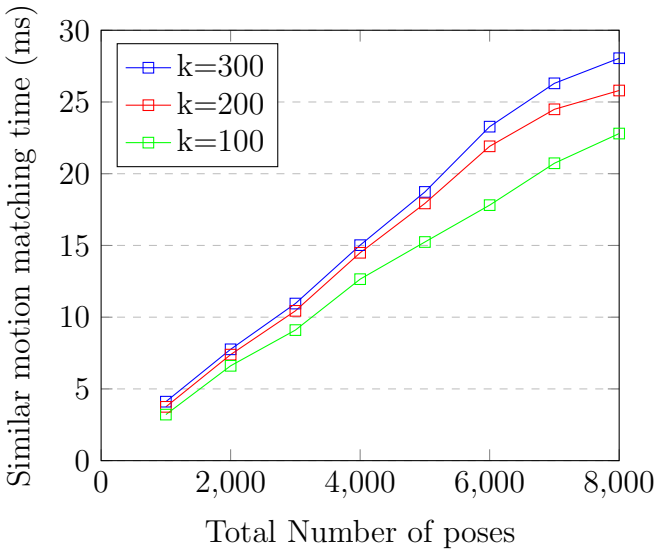


Figure 4.5: Similar motion matching time (ms)/Total number of poses

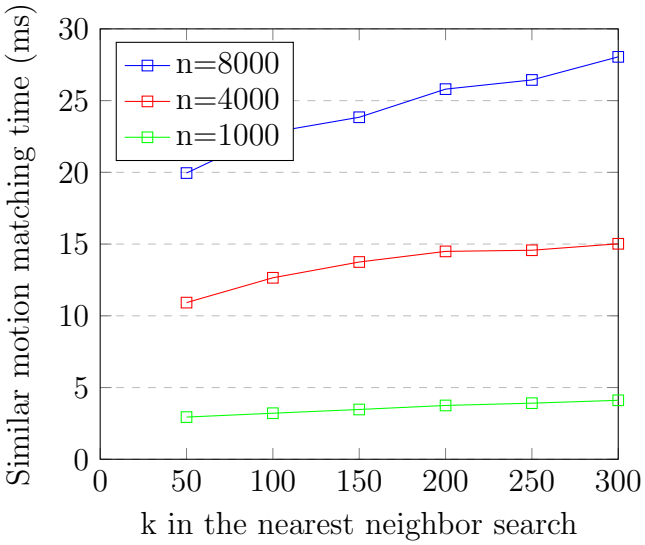


Figure 4.6: Similar motion matching time (ms)/k in the nearest neighbor search

4.2.2 Results

We use precision and recall metrics to demonstrate the relevance of our query results. Precision is the percentage of retrieved results that are relevant and

Recall is the percentage of relevant results that are retrieved [16]. Precision is computed using Equation 4.1 and recall is computed using Equation 4.2. The terms that are used in Equations 4.1 and 4.2 are shown on Table 4.4 [16].

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

	Relevant	Nonrelevant
Retrieved	true positives (TP)	false positives (FP)
Not retrieved	false negatives (FN)	true negatives (TN)

Table 4.4: The terms used in Equations 4.1 and 4.2

Our dataset has 95 motion files that consists of 1231 $\times$ 60 frames. The motion types in the database are jumping, walking, running, ball dribbling, punching, kicking football and various dance movements. The p value is the half of the frame rate (FPS/2), which is equal to 60. n is equal to 1231.

The TP, FP, TN and FP values of nearest neighbor searches of three sample motions of three different motion types on this database are shown on Table 4.5.

	jump				walk				run			
	TP	FP	TN	FN	TP	FP	TN	FN	TP	FP	TN	FN
k=50	2	0	83	10	4	0	64	27	6	0	78	11
k=100	3	0	83	9	8	0	64	23	7	0	78	10
k=150	4	0	83	8	11	0	64	20	8	1	77	9
k=200	5	0	83	7	14	0	64	17	8	6	72	9
k=250	8	0	83	4	14	3	61	17	9	12	66	8

Table 4.5: TP, FP, TN and FP values of the three kind of motions for various k values

The precision rates of three sample motions of three different motion types are shown on Table 4.6. Our approach gives good precision results unless k value is high. This shows that most of the returned motions are relevant unless k value is high.

	jump	walk	run
k=50	2/2	4/4	6/6
k=100	3/3	8/8	7/7
k=150	4/4	11/11	8/9
k=200	5/5	14/14	8/14
k=250	8/8	14/17	9/21

Table 4.6: Precision rates of jump, walk and run motion types for various k values.

The recall rates of sample motions of three different motion types are shown on Table 4.7. As expected, increase in the k value, increases the number of resulting motions from the k nearest neighbors query. However, our approach cannot retrieve all relevant motions even if k is 250 because of the existence of variations in the time domain between different motion files in the same category. It is not possible to handle variations in the time domain unless variation and inexactness algorithms of Müller et al. are implemented. They can be applied to our hybrid approach because we use the motion matching technique of Müller et al.

	jump	walk	run
k=50	2/12	4/31	6/17
k=100	3/12	8/31	7/17
k=150	4/12	11/31	8/17
k=200	5/12	14/31	8/17
k=250	8/12	14/31	9/17

Table 4.7: Recall rates of jump, walk and run motion types for various k values.

4.2.3 Visual Results

The snapshots in this section are taken from downloadable videos of corresponding motion files of CMU Motion Capture Database [1].

4.2.3.1 Jump

Figure 4.7 gives the snapshots of the query jumping motion. Figures 4.8 and 4.9 give the snapshots of the resulting motions for k=100, whose TP, FP, TN and FN values are given in Table 4.5.

Figure 4.7: Query jumping motion

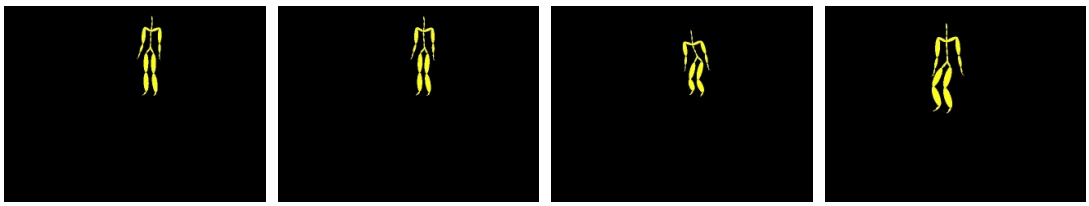


Figure 4.8: Resulting jumping motion 1

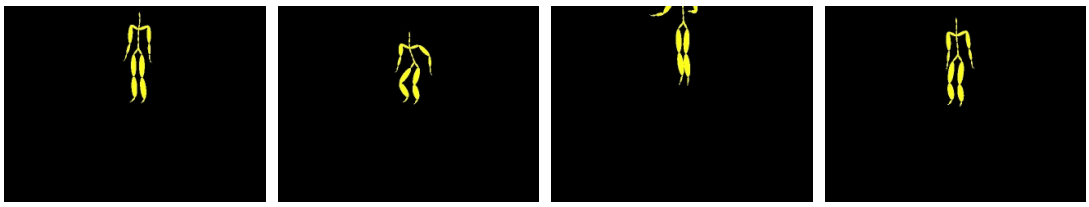


Figure 4.9: Resulting jumping motion 2

4.2.3.2 Walk

Figure 4.10 gives the snapshots of the query walking motion. Figures 4.11, . . . , 4.17 give the snapshots resulting motions for k=100, whose TP, FP, TN and FN values are given in Table 4.5.

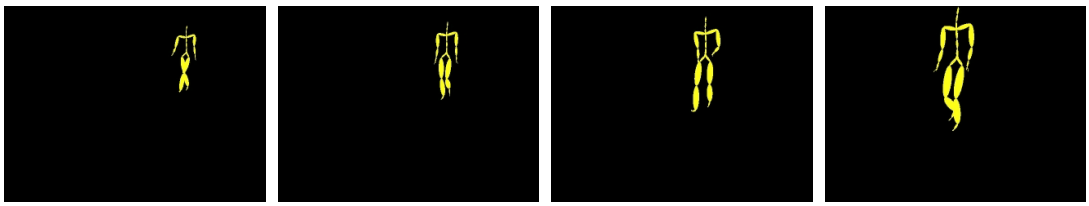


Figure 4.10: Query walking motion

Figure 4.11: Resulting walking motion 1

Figure 4.12: Resulting walking motion 2

Figure 4.13: Resulting walking motion 3

Figure 4.14: Resulting walking motion 4

Figure 4.15: Resulting walking motion 5

Figure 4.16: Resulting walking motion 6

Figure 4.17: Resulting walking motion 7

4.2.3.3 Run

Figure 4.18 gives the snapshots of the running query motion. Figure 4.19, ... , 4.24 give the snapshots of resulting motions for k=100, whose TP, FP, TN and FN values are given in Table 4.5.

Figure 4.18: Query running motion

Figure 4.19: Resulting running motion 1

Figure 4.20: Resulting running motion 2

Figure 4.21: Resulting running motion 3

Figure 4.22: Resulting running motion 4

Figure 4.23: Resulting running motion 5

Figure 4.24: Resulting running motion 6

Chapter 5

Conclusion

In this thesis, we proposed a hybrid approach that utilizes numerical based feature sets and k-d tree [6] based indexing structure of Krüger et al. [5] and inverted index based motion matching technique of Müller et al. [3]. Our hybrid approach does not need user input, can be used in environments requiring close similarity of motions and can be used with variation and inexactness algorithms of Müller et al.

For pose matching, we used the feature set ${F_E}^{15}$ for human motion data for indexing and searching poses using k-d tree structure. Building our indexing structure has a time complexity of $O(nlg(n))$ where n is number of poses in the database. Searching k nearest neighbors of a pose in the database is $O(klg(n))$. Our performance and query results showed that ${F_E}^{15}$ can be used for searching and indexing activities for practical use.

In the motion matching part, we found nearest neighbors of poses for the given query motion. We used inverted indexes and basic set operations to find exact hits efficiently. Our precision and recall results for motion matching showed that our hybrid approach can be used to retrieve similar motions unless there are significant variations in the time domain are present.

However, our approach enables implementation of time variation handling and

inexactness algorithms of Müller et al. These algorithms are named adaptive segmentation and k-mismatch search, respectively. The implementation of these algorithms can be considered as future work. Another idea for future work could be using the subsets of ${F_E}^{15}$ for much coarsened queries.

Chapter 6

Bibliography

[1] "Carnegie-Mellon Mocap Database." http://mocap.cs.cmu.edu, 2003.

[2] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber, "Documentation Mocap Database HDM05," *Database*, no. CG-2007-2, p. 34, 2007.

[3] M. Müller and T. Röder, "Motion templates for automatic classification and retrieval of motion capture data," *Proceedings of '06 ACM SIGGRAPH*, pp. 137–146, 2006.

[4] M. Clausen and F. Kurth, "A Unified Approach to Content-Based and Fault-Tolerant Music Recognition," *IEEE Transactions on Multimedia*, vol. 6, pp. 717–731, Oct. 2004.

[5] B. Krüger, J. Tautges, A. Weber, and A. Zinke, "Fast Local and Global Similarity Searches in Large Motion Capture Databases," in *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pp. 1–10, 2010.

[6] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[7] M. Müller, "Dynamic time warping," in *Information Retrieval for Music and Motion*, pp. 69–84, Springer Berlin Heidelberg, 2007.

[8] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[9] M. Cardle, M. Vlachos, and S. Brooks, "Fast motion capture matching with replicated motion editing," *Proceedings of ACM SIGGRAPH '03 Sketches*, 2003.

[10] C.-Y. Chiu, S.-P. Chao, M.-Y. Wu, S.-N. Yang, and H.-C. Lin, "Content-based retrieval for human motion data," *Journal of Visual Communication and Image Representation*, vol. 15, pp. 446–466, Sept. 2004.

[11] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle, "Indexing Large Human-Motion Databases," in *Proceedings of the 30th International Conference on Very Large Data Bases - Volume 30*, pp. 780–791, 2004.

[12] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large data sets," *Proceedings of ACM SIGGRAPH '04*, vol. 3, p. 559, 2004.

[13] J. Chai and J. K. Hodgins, "Performance animation from low-dimensional control signals," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 686–696, 2005.

[14] "Acclaim ASF/AMC." http://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ASF-AMC.html, 1999.

[15] D. Tolani, A. Goswami, and N. I. Badler, "Real-time inverse kinematics techniques for anthropomorphic limbs.," *Graphical Models*, vol. 62, no. 5, pp. 353–388, 2000.

[16] Manning, Christopher D. and Raghavan, Prabhakar and Schütze, Hinrich and others, *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

Appendix A

Appendix

A.1 Motion Files

axis, *order*, *position* and *orientation* values of the root bone are defined in *root* section of the ASF File. *id*, *name*, *direction*, *length*, *axis*, *dof* and *limit* values of each bone are defined in the bonedata section of the ASF file. All of the bones are defined relative to the root bone. These values defines a base pose as a starting point for the motion data. A sample ASF file shown in Figure A.1.

In the AMC file, bone values for each frame start with the corresponding frame number. Transformation values are defined for each joint. A sample AMC file shown in Figure A.2.

A.2 Motion Viewer

The screenshot of our motion viewer program that plays query and results motions in real time is given in Figure A.3. The motion on the left is the query and the motion on the right is one of the results. Results can be browsed through keyboard keys.

```
:root
   axis XYZ
   order TX TY TZ RZ RY RX
   position 0.0 0.0 0.0
   orientation 0.0 0.0 0.0
:bonedata
begin
   id 1
   name hips
   direction 0.000000 1.000000 0.000000
   length 0.000000
   axis 0.00000 0.00000 0.00000 XYZ
   dof rx ry rz
   limits (-180.0 180.0)
      (-180.0 180.0)
      (-180.0 180.0)
end
begin
   id 2
   name hips1
   direction 0.000000 1.000000 0.000000
   length  4.310000
   axis 0.000 0.000 0.000 XYZ
end
begin
   id 3
   name chest
   direction 0.000000 1.000000 0.000000
   length 0.000000
   axis 0.00000 0.00000 0.00000 XYZ
   dof rx ry rz
   limits (-180.0 180.0)
      (-180.0 180.0)
      (-180.0 180.0)
end
```

Figure A.1: A sample ASF file

```
1
root -1.244205 36.710186 -1.148101 0.958161 4.190043 -18.282991
hips 0.000000 0.000000 0.000000
chest 15.511776 -2.804996 -0.725314
neck 48.559605 0.000000 0.014236
head -38.332661 1.462782 -1.753684
leftcollar 0.000000 15.958783 0.921166
leftuparm -10.319685 -15.040003 63.091194
leftlowarm -27.769176 -15.856658 8.187016
lefthand 2.601753 -0.217064 -5.543770
rightcollar 0.000000 -8.470076 2.895008
rightuparm 6.496142 9.551583 -57.854118
rightlowarm -26.983490 11.338276 -5.716377
righthand -6.387745 -1.258509 5.876069
leftupleg 23.412262 -5.325913 12.099395
leftlowleg -6.933442 -6.276054 -1.363996
leftfoot -1.877641 4.455667 -6.275022
rightupleg 20.698696 3.189690 -8.377244
rightlowleg 3.445840 -6.717122 2.046032
rightfoot -8.162314 0.687809 9.000264
2
root -0.227361 37.620358 1.672587 0.204373 -4.264866 -12.155879
hips 0.000000 0.000000 0.000000
chest 14.747641 2.858763 -1.345236
neck 44.651531 0.000000 -0.099206
head -38.546989 0.678145 -4.633668
leftcollar 0.000000 7.233337 -5.791124
leftuparm 9.928153 -50.015823 25.218475
leftlowarm -40.443512 -0.566324 0.482702
lefthand 6.011584 -0.216811 4.576208
rightcollar 0.000000 -1.936009 5.471129
rightuparm 3.926107 32.418419 -26.396805
rightlowarm -43.958717 3.548671 -3.415734
righthand -4.901258 -0.112565 0.681468
leftupleg 11.932759 0.406248 -1.921313
leftlowleg 13.698170 5.503362 2.643481
leftfoot -16.237123 2.755839 -7.182952
rightupleg 13.767217 0.331739 -1.353482
rightlowleg 22.576195 -7.388037 -3.537788
rightfoot -19.946142 2.525145 8.668705
                                                                    ]
```

Figure A.2: A sample AMC file

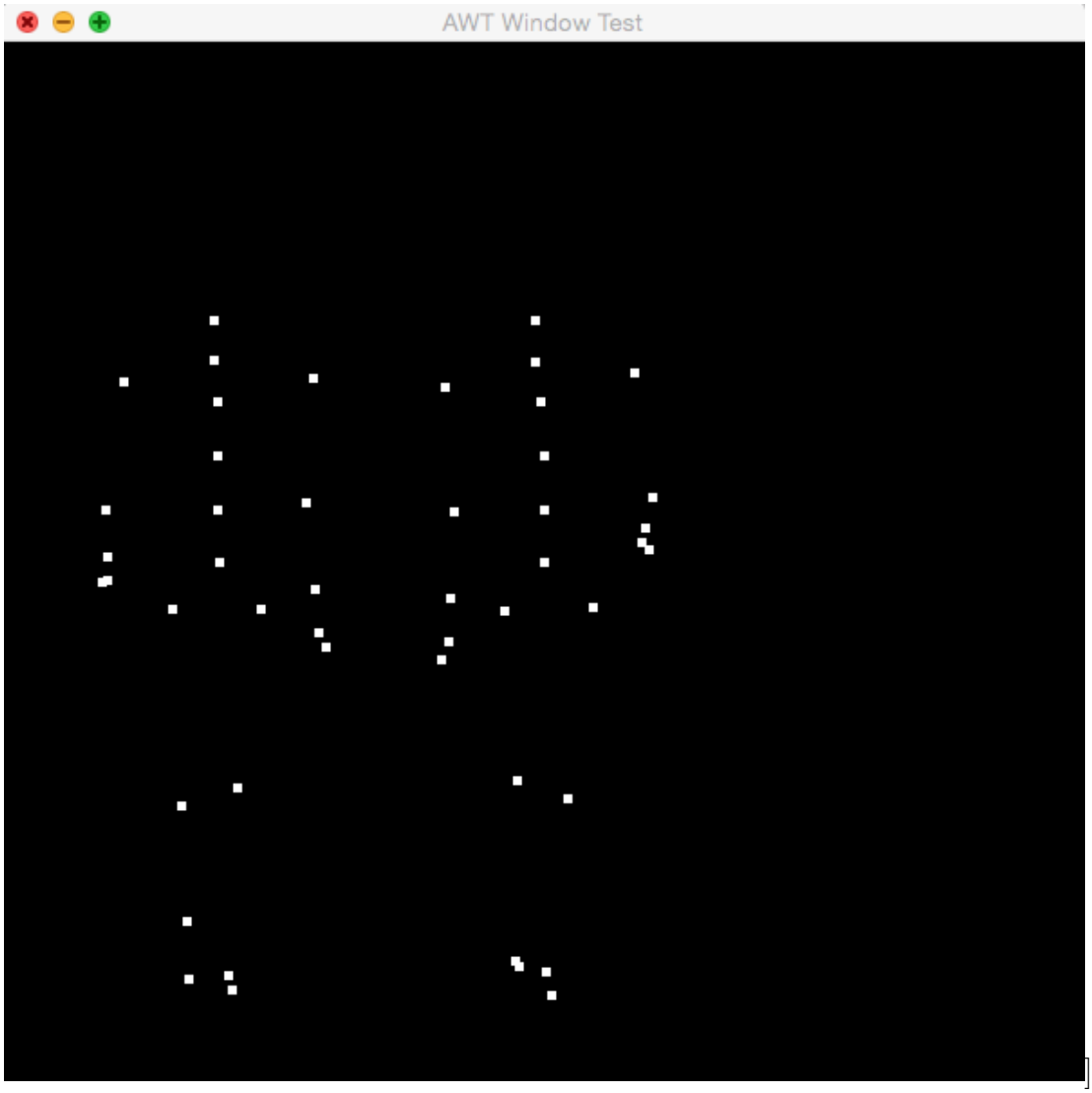


Figure A.3: The screenshot of our motion viewer program