FMDistance: A fast and effective distance function for motion capture data

Kensuke Onuma[*] **Christos Faloutsos**[†]
Jessica K. Hodgins[†]

April 2008
CMU-CS-07-164

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[*]Sony Corporation, Tokyo, Japan
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Given several motion capture sequences, of similar (but not identical) length, what is a good distance function? We want to find similar sequences, to spot outliers, to create clusters, and to visualize the (large) set of motion capture sequences at our disposal. We propose a set of new features for motion capture sequences. We experiment with numerous variations (112 feature-sets in total, using variations of weights, logarithms, dimensionality reduction), and we show that the appropriate combination leads to near-perfect classification on a database of 226 actions with twelve different categories, and it enables visualization of the whole database as well as outlier detection.

Keywords: data mining, motion capture, distance function, classification, human animation

1 Introduction

Motion capture data is often used to create human animations for video games, movies and other applications. Large databases of motion now exist both on the web (see `http://mocap.cs.cmu.edu/`, for example) and as proprietary resources within entertainment companies. These databases are not easily searchable. In this paper, we present a distance function that represents the characteristics of the actions in a motion capture sequence. This distance metric is suitable for classifying motions, searching for similar motions, and for detecting outlier motions.

We would like a distance function that is both fast and effective. The distance function should be fast to compute, even on long motion capture sequences. Ideally, it should be independent ($O(1)$) on the length N_{frame} of the sequences. The distance function should be also meaningful, so that it is useful for clustering, classification, and anomaly detection (see Figure 1).

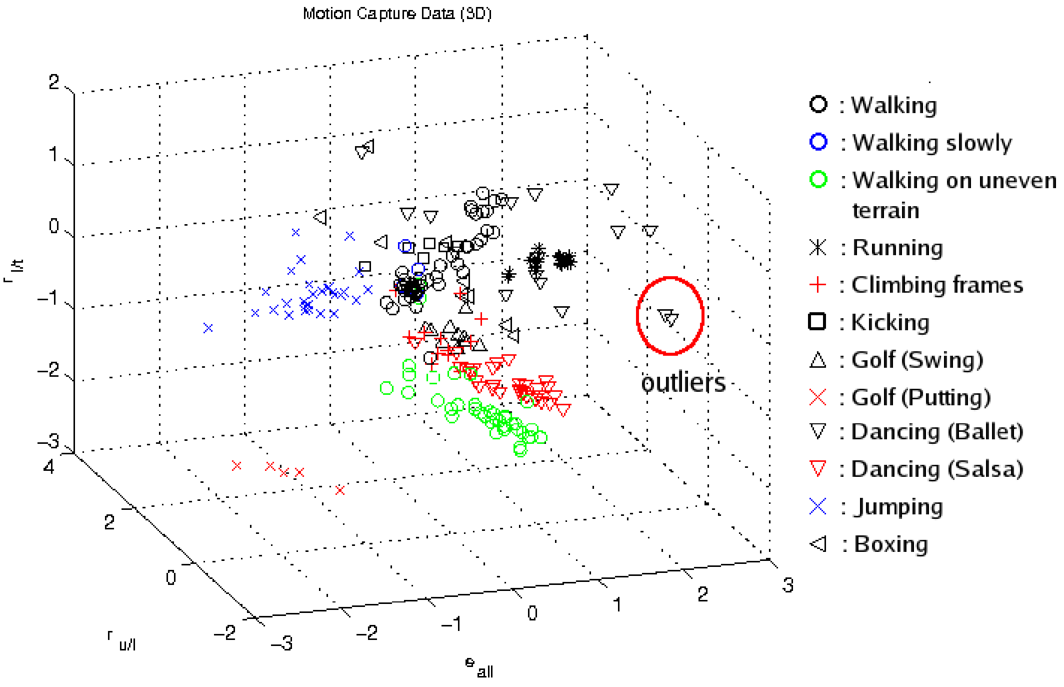


Figure 1: Visualization of classification by three new features. Notice the *homophily*: same motions cluster together; also notice that we can visually spot outliers, like ballet dancing which has noisy capture frames, in the red circle.

Specifically, we propose $FMDistance$, a method which is *independent* on the sequence length, and only depends on K, the number of joint angles we track. Our idea is to calculate the approximation of the total kinetic energy of each joint as a preprocessing step, thus compressing each motion capture sequence of $K \times N_{frame}$ numbers into K numbers (and possibly, even fewer, if we do dimensionality reduction). The proposed method is also effective, as we illustrate by results of the accuracy of motion classification and visualization of the relation among motion capture data in Section 4.

The rest of the paper is organized as follows: we first review the related work in Section 2; the proposed strategies are presented in Section 3; the experimental results are presented in Section 4; and we conclude the paper in Section 5.

2 Related work

To our knowledge, there is little work in computer graphics that focuses on a distance function for expressing the characteristics of a motion capture sequence. The work of Ren et al.[7], who explored methods for verifying the naturalness of a motion capture sequence, is close to our problem. Troje [9] investigated gender classification of walking motions by analyzing motion capture data. Researchers in computer vision also proposed distance functions for human activity classification [1].

Many distance functions have been proposed for finding candidates of "frame to frame" transition [3, 5, 6, 8, 10] or indexing for segmentation [4]. However, the distance function in [3, 5, 6, 8, 10] requires $K \times N_{frame}$ computations; Our proposed method does not depend on N_{frame} and is much simpler and faster ($O(K)$).

3 Proposed method

In this section, we describe details of $FMDistance$ for the classification of motion capture data. First we describe the way to make the data independent of time in preprocessing step, and then we introduce how to calculate the distance between two motion capture sequences.

3.1 Preprocessing step : Transformation from motion capture data to kinetic energy-based parameters

We assume that motion capture data have K DOF in total and a series of motion capture data for one action has N_{frame} frames. Specifically, every frame has joint angles, the root orientation and the root positions (coordinates of the root). A set of motion capture data is denoted by $\{\vec{x}_i | i = 1, ..., N_{frame}\}$. Each frame $\vec{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,K}]^T$ represents a point in K dimensional space.

The main idea is to use the average of the approximate kinetic energy of each joint angle (or carefully selected groups of joints), as features. This way, the $K \times N_{frame}$ numbers of a motion capture sequence are condensed into at most K values as a preprocessing step, achieving our first goal, speed.

We compute the approximate kinetic energy as the sum of squares of velocities. Specifically, the velocity of the root and the angular velocity of each joint, $\vec{v}_i = [v_{i,1}, v_{i,2}, ..., v_{i,K-1}]^T$ are calculated by the first derivatives:

$$\vec{v}_i = \frac{\vec{x}_{i+1} - \vec{x}_i}{t_{frame}} \tag{1}$$

where t_{frame} is the period between frames. For the root velocity, we calculate the velocity and the energy across the plane from the x + z position. Thus, the dimensionality of $\vec{v}_i$ decreases by 1, to $K-1$. We note that the vertical velocity of the root is included as one of parameters.

To compute the kinetic energy, we also need to consider the moment of inertia m_j of each joint, (and body mass, for each position-coordinate). The kinetic energy $E_{i,j}$ of joint j at time i is

$$E_{i,j} = m_j \times v_{i,j}^2 \tag{2}$$

Although the moment of inertia varies depending on the body part, we assume that they are constant with respect to time.

3.2 Distance function for classification

We calculate the mean of the kinetic energy at each dimension (joint angle/position/orientation), j:

$$\overline{E}_j \; = \; \frac{1}{N_{frame}} \sum_{i=1}^{N_{frame}} E_{i,j} \tag{3}$$

The kinetic energy is bursty: some joints have a high kinetic energy while others do not. We propose to treat the burstiness, by taking logarithms, specifically $\log(x+1)$ (to handle the joints of zero energy). Thus:

$$e_j \; = \; \log\left(\overline{E}_j + 1\right) \tag{4}$$

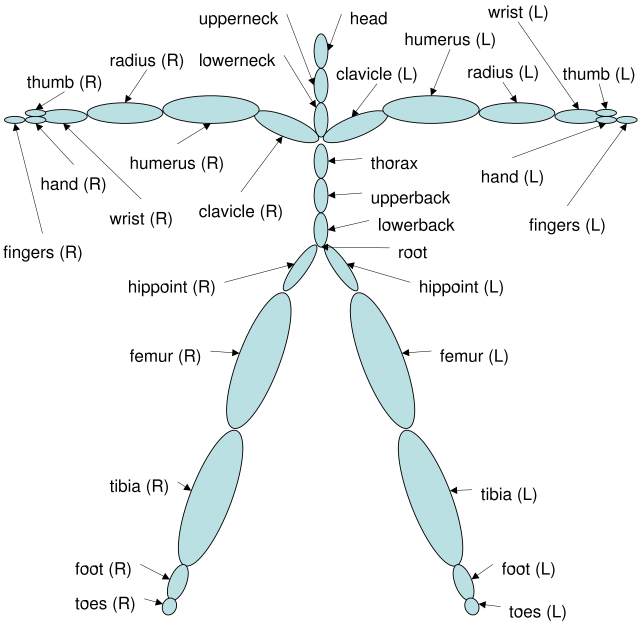
The vector $\vec{e} = [e_1, e_2, ..., e_{K-1}]^T$ is our proposed feature vector. Then, the distance between two motion capture sequences N and M is the Euclidean distance of their feature vectors $\vec{e}_N$ and $\vec{e}_M$.

4 Experimental results

In this section, we evaluate the effectiveness of our approach. The motion capture data we use for the experiments is `http://mocap.cs.cmu.edu/`. Figure 2 shows the human figure, the number of DOF, and the value of m_j for each joint. The value of m_j heuristically models the moment of inertia: hip joints get high values, shoulders get a bit smaller, knees are next, elbows are next, etc.

We used 226 sequences of motion capture data and they are categorized into the twelve actions described in Figure 6. We also carefully examined 112 feature sets, from the cross products of $[\{\log, \text{lin}\} \times \{\text{original data, transformed data from quaternion}\} \times \{\text{weighted } m_j, \text{constant } m_j\} \times \{\text{normalized, unnormalized}\} \times \{\text{seven feature sets}\}]$.

Implementation details: The distance function for each feature set was the Euclidean distance. Before calculating velocities, we followed standard practice and removed noise with a linear low-pass filter spanning five frames. Finally, when we normalized the data in order that they have zero mean and unit standard deviation.



bone	DOF	m_j
root (pos)	3	1
root (rotation)	3	5
hippoint	0	0
lowerback	3	4
upperback	3	4
thorax	3	4
lowerneck	3	3
upperneck	3	3
head	3	0
clavicle	2	3
humerus	3	3
radius	1	2
wrist	1	1
hand	2	0
fingers	1	0
thumb	2	0
femur	3	3
tibia	1	3
foot	2	0
toes	1	0

Figure 2: Human figure model, DOFs and the distribution of m_j. It has 29 joints, 56 joint angles, three angles for the root orientation and three position coordinates for the root.

Effectiveness measure: To measure the effectiveness of our feature sets, we use the classification accuracy, and specifically, a 1-nearest neighbor (1-NN) classifier. We chose this classifier because its accuracy is directly related to the effectiveness of the feature set, and it needs no training. Moreover, as we show next, it gives excellent classification accuracy.

4.1 Accuracy of various feature sets

We use the feature sets described in Section 3, as well as some other, *simpler* feature sets, for comparison. The nomenclature for a feature set is as follows: For example, *61-LOG-cons-qua-norm* stands for 61 features, with the log transform, constant values for the moments/weights m_j, converted by quaternion, and normalized. Similarly, *61-LIN-est-nqua-unnorm* stands for the same 61 features, without the log transform, with the estimated values of the m_j weights as shown in Figure 2, not converted by quaternion, and unnormalized. Figure 3 gives the list of feature sets we tried, and their descriptions.

Tables 1-4 show results of our experiments. They show that 61-LOG-cons-nqua-norm is the best feature set for classifying motion capture data (the accuracy is same as 62-LOG-cons-nqua-norm, but we take the feature set with fewer parameters).

3-LOG The features shown in Equation (6).

3-LIN The average of the total approximate kinetic energy, the ratio of the approximate kinetic energy between upper body vs lower body, limbs vs trunk ($\overline{E}_{total}, \frac{\overline{E}_{upper}}{\overline{E}_{lower}}, \frac{\overline{E}_{limbs}}{\overline{E}_{trunk}}$).

4-LOG The features shown in Equation (6) plus the logarithm of the ratio of the approximate kinetic energy between right body and left body ($r_{r/l} = \log\left(\frac{\overline{E}_{right}+1}{\overline{E}_{left}+1}\right)$).

4-LIN The average of the total approximate kinetic energy, the ratio of the approximate kinetic energy between upper body vs lower body, limbs vs trunk, and right body vs left body ($\overline{E}_{total}, \frac{\overline{E}_{upper}}{\overline{E}_{lower}}, \frac{\overline{E}_{limbs}}{\overline{E}_{trunk}}, \frac{\overline{E}_{right}}{\overline{E}_{left}}$).

29-LOG The logarithms of the mean of the approximate kinetic energy for each joint.

29-LIN The mean of the approximate kinetic energy for each joint.

31-LOG The logarithms of the root's energy in the horizontal and vertical direction, as well as of the approximate kinetic energy of each joint.

31-LIN The root's energy in the horizontal and vertical direction, as well as the approximate kinetic energy of each joint.

59-LOG Log of the mean of the approximate kinetic energy for each joint angle (not including kinetic energy of root).

59-LIN Mean of the approximate kinetic energy for each joint angle (not including kinetic energy of root).

61-LOG Log of the mean of the approximate kinetic energy for each joint angle plus vertical and horizontal kinetic energy of root. See Equation (4).

61-LIN Mean of the approximate kinetic energy for each joint angle plus vertical and horizontal kinetic energy of root. See Equation (3).

62-LOG Log of the mean of the approximate kinetic energy for each joint angle plus kinetic energy of root for x, y, z axes.

62-LIN Mean of the approximate kinetic energy for each joint angle plus kinetic energy of root for x, y, z axes.

62-POS The mean of $\vec{x}_i$ with respect to time.

Figure 3: Feature sets: names and descriptions

features	m_j	quaternion	normalized	% error
3-LOG-cons-nqua-norm	constant	N	Y	16.37%
3-LOG-est-nqua-norm	estimated	N	Y	11.50%
3-LIN-cons-nqua-norm	constant	N	Y	22.57%
3-LIN-est-nqua-norm	estimated	N	Y	16.81%
3-LOG-cons-qua-norm	constant	Y	Y	19.47%
3-LOG-est-qua-norm	estimated	Y	Y	15.93%
3-LIN-cons-qua-norm	constant	Y	Y	21.68%
3-LIN-est-qua-norm	estimated	Y	Y	22.12%
3-LOG-cons-nqua-unnorm	constant	N	N	16.37%
3-LOG-est-nqua-unnorm	estimated	N	N	11.95%
3-LIN-cons-nqua-unnorm	constant	N	N	29.20%
3-LIN-est-nqua-unnorm	estimated	N	N	37.17%
3-LOG-cons-qua-unnorm	constant	Y	N	21.68%
3-LOG-est-qua-unnorm	estimated	Y	N	16.37%
3-LIN-cons-qua-unnorm	constant	Y	N	34.96%
3-LIN-est-qua-unnorm	estimated	Y	N	29.20%
4-LOG-cons-nqua-norm	constant	N	Y	15.49%
4-LOG-est-nqua-norm	estimated	N	Y	10.62%
4-LIN-cons-nqua-norm	constant	N	Y	20.80%
4-LIN-est-nqua-norm	estimated	N	Y	17.70%
4-LOG-cons-qua-norm	constant	Y	Y	15.04%
4-LOG-est-qua-norm	estimated	Y	Y	14.60%
4-LIN-cons-qua-norm	constant	Y	Y	23.45%
4-LIN-est-qua-norm	estimated	Y	Y	20.80%
4-LOG-cons-nqua-unnorm	constant	N	N	13.27%
4-LOG-est-nqua-unnorm	estimated	N	N	9.73%
4-LIN-cons-nqua-unnorm	constant	N	N	29.20%
4-LIN-est-nqua-unnorm	estimated	N	N	37.17%
4-LOG-cons-qua-unnorm	constant	Y	N	15.49%
4-LOG-est-qua-unnorm	estimated	Y	N	12.83%
4-LIN-cons-qua-unnorm	constant	Y	N	34.51%
4-LIN-est-qua-unnorm	estimated	Y	N	29.65%

Table 1: Feature sets and their classification error - part A

features	m_j	quaternion	normalized	% error
29-LOG-cons-nqua-norm	constant	N	Y	4.87%
29-LOG-est-nqua-norm	estimated	N	Y	4.42%
29-LIN-cons-nqua-norm	constant	N	Y	8.85%
29-LIN-est-nqua-norm	estimated	N	Y	5.31%
29-LOG-cons-qua-norm	constant	Y	Y	4.42%
29-LOG-est-qua-norm	estimated	Y	Y	5.31%
29-LIN-cons-qua-norm	constant	Y	Y	7.52%
29-LIN-est-qua-norm	estimated	Y	Y	4.42%
29-LOG-cons-nqua-unnorm	constant	N	N	3.98%
29-LOG-est-nqua-unnorm	estimated	N	N	3.98%
29-LIN-cons-nqua-unnorm	constant	N	N	8.85%
29-LIN-est-nqua-unnorm	estimated	N	N	15.04%
29-LOG-cons-qua-unnorm	constant	Y	N	4.87%
29-LOG-est-qua-unnorm	estimated	Y	N	5.31%
29-LIN-cons-qua-unnorm	constant	Y	N	9.29%
29-LIN-est-qua-unnorm	estimated	Y	N	10.62%
31-LOG-cons-nqua-norm	constant	N	Y	3.10%
31-LOG-est-nqua-norm	estimated	N	Y	3.10%
31-LIN-cons-nqua-norm	constant	N	Y	6.19%
31-LIN-est-nqua-norm	estimated	N	Y	4.42%
31-LOG-cons-qua-norm	constant	Y	Y	3.10%
31-LOG-est-qua-norm	estimated	Y	Y	3.98%
31-LIN-cons-qua-norm	constant	Y	Y	6.64%
31-LIN-est-qua-norm	estimated	Y	Y	3.98%
31-LOG-cons-nqua-unnorm	constant	N	N	4.87%
31-LOG-est-nqua-unnorm	estimated	N	N	3.54%
31-LIN-cons-nqua-unnorm	constant	N	N	11.95%
31-LIN-est-nqua-unnorm	estimated	N	N	15.04%
31-LOG-cons-qua-unnorm	constant	Y	N	3.98%
31-LOG-est-qua-unnorm	estimated	Y	N	3.10%
31-LIN-cons-qua-unnorm	constant	Y	N	12.39%
31-LIN-est-qua-unnorm	estimated	Y	N	11.95%

Table 2: Feature sets and their classification error - part B

features	m_j	quaternion	normalized	% error
59-LOG-cons-nqua-norm	constant	N	Y	3.54%
59-LOG-est-nqua-norm	estimated	N	Y	3.10%
59-LIN-cons-nqua-norm	constant	N	Y	5.75%
59-LIN-est-nqua-norm	estimated	N	Y	4.87%
59-LOG-cons-qua-norm	constant	Y	Y	4.42%
59-LOG-est-qua-norm	estimated	Y	Y	2.65%
59-LIN-cons-qua-norm	constant	Y	Y	5.75%
59-LIN-est-qua-norm	estimated	Y	Y	4.42%
59-LOG-cons-nqua-unnorm	constant	N	N	3.98%
59-LOG-est-nqua-unnorm	estimated	N	N	3.54%
59-LIN-cons-nqua-unnorm	constant	N	N	7.96%
59-LIN-est-nqua-unnorm	estimated	N	N	13.27%
59-LOG-cons-qua-unnorm	constant	Y	N	4.42%
59-LOG-est-qua-unnorm	estimated	Y	N	3.54%
59-LIN-cons-qua-unnorm	constant	Y	N	10.62%
59-LIN-est-qua-unnorm	estimated	Y	N	4.87%
61-LOG-cons-nqua-norm	constant	N	Y	**2.21%**
61-LOG-est-nqua-norm	estimated	N	Y	2.65%
61-LIN-cons-nqua-norm	constant	N	Y	3.98%
61-LIN-est-nqua-norm	estimated	N	Y	4.42%
61-LOG-cons-qua-norm	constant	Y	Y	3.54%
61-LOG-est-qua-norm	estimated	Y	Y	2.65%
61-LIN-cons-qua-norm	constant	Y	Y	4.87%
61-LIN-est-qua-norm	estimated	Y	Y	3.98%
61-LOG-cons-nqua-unnorm	constant	N	N	3.10%
61-LOG-est-nqua-unnorm	estimated	N	N	2.65%
61-LIN-cons-nqua-unnorm	constant	N	N	13.27%
61-LIN-est-nqua-unnorm	estimated	N	N	16.37%
61-LOG-cons-qua-unnorm	constant	Y	N	2.65%
61-LOG-est-qua-unnorm	estimated	Y	N	2.65%
61-LIN-cons-qua-unnorm	constant	Y	N	13.72%
61-LIN-est-qua-unnorm	estimated	Y	N	12.39%

Table 3: Feature sets and their classification error - part C. The feature set with the lowest classification error, 61-LOG-cons-nqua-norm, is in bold.

features	m_j	quaternion	normalized	% error
62-LOG-cons-nqua-norm	constant	N	Y	2.21%
62-LOG-est-nqua-norm	estimated	N	Y	2.65%
62-LIN-cons-nqua-norm	constant	N	Y	4.87%
62-LIN-est-nqua-norm	estimated	N	Y	4.42%
62-LOG-cons-qua-norm	constant	Y	Y	3.54%
62-LOG-est-qua-norm	estimated	Y	Y	2.65%
62-LIN-cons-qua-norm	constant	Y	Y	4.87%
62-LIN-est-qua-norm	estimated	Y	Y	3.54%
62-LOG-cons-nqua-unnorm	constant	N	N	3.10%
62-LOG-est-nqua-unnorm	estimated	N	N	3.10%
62-LIN-cons-nqua-unnorm	constant	N	N	12.83%
62-LIN-est-nqua-unnorm	estimated	N	N	15.49%
62-LOG-cons-qua-unnorm	constant	Y	N	3.10%
62-LOG-est-qua-unnorm	estimated	Y	N	3.54%
62-LIN-cons-qua-unnorm	constant	Y	N	14.60%
62-LIN-est-qua-unnorm	estimated	Y	N	12.83%
62-POS-norm	N/A	N/A	Y	5.75%
62-POS-unnorm	N/A	N/A	N	20.80%

Table 4: Feature sets and their classification error - part D (last).

We report the conclusions and observations:

- Figure 4 illustrates how the features improve. The classification error reduces steeply at the lower rank features and it still declines gradually at the higher rank features. It implies that there might still have a space to increase classification accuracy.
- Taking the logarithms ($\log(x+1)$) almost always improves performance. Figure 5 provides the result that all features, except one feature (*29-***-est-qua-norm*), which take the logarithm have lower classification error than those which do not take the logarithm.
- There is no clear winner with respect to the sets of m_j weights. Both ”cons” and ”est” perform about the same.

Figure 6 shows the *confusion matrix* for our best performer, the *61-LOG-cons-nqua-norm* feature set. Columns correspond to the predicted labels, and rows to the actual label. In a perfect classifier, the matrix would be diagonal. Notice that *61-LOG-cons-nqua-norm* gives a near-diagonal matrix. The sequences it confused were all “Walking” sequences, (“Walking”, “Walking slowly”, and “Walking on uneven terrain”).

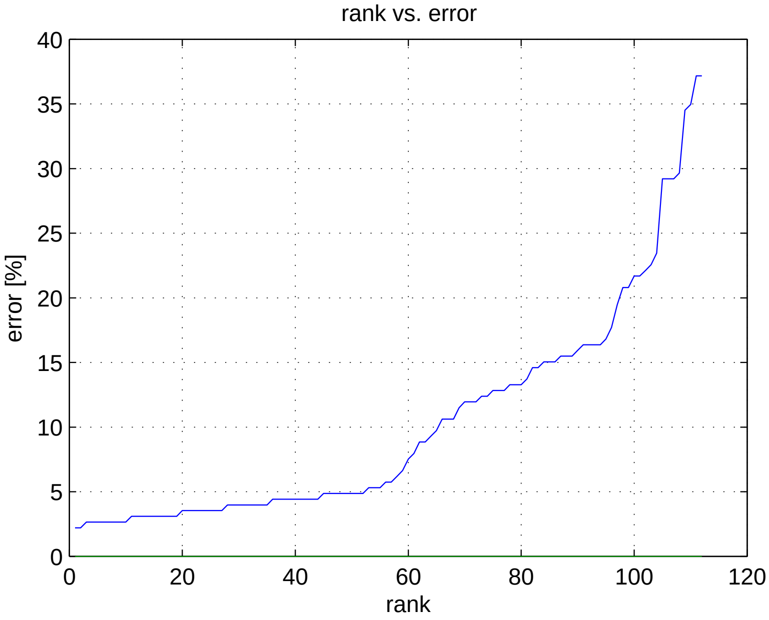


Figure 4: The distribution of the classification error among feature set. X-axis represents a rank of the features sorted by percentage of the classification error and y-axis means percentage of the classification error.

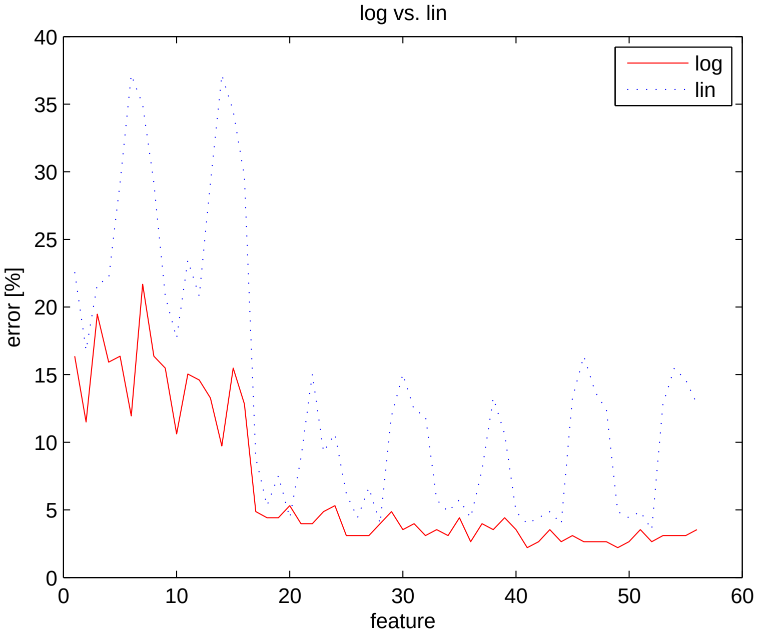
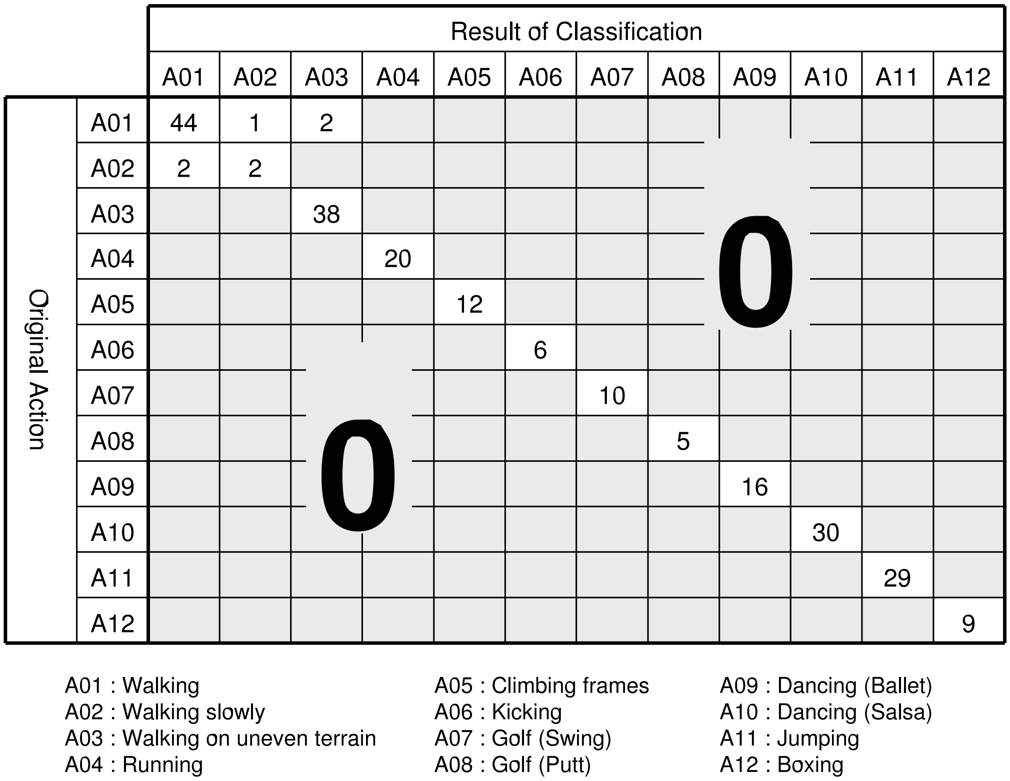


Figure 5: Comparison of the classification error by taking the logarithm and that by not taking the logarithm.



		Result of Classification											
		A01	A02	A03	A04	A05	A06	A07	A08	A09	A10	A11	A12
Original Action	A01	44	1	2									
	A02	2	2										
	A03			38									
	A04				20								
	A05					12							
	A06						6						
	A07							10					
	A08								5				
	A09									16			
	A10										30		
	A11											29	
	A12												9

A01 : Walking
A02 : Walking slowly
A03 : Walking on uneven terrain
A04 : Running
A05 : Climbing frames
A06 : Kicking
A07 : Golf (Swing)
A08 : Golf (Putt)
A09 : Dancing (Ballet)
A10 : Dancing (Salsa)
A11 : Jumping
A12 : Boxing

Figure 6: Confusion Matrix of condition 61-LOG-cons-nqua-norm in Table 4. Notice that (a) it is near-diagonal and (b) the confused motions are very similar ($A01$-$A03$), all being variations of walking motions.

We also examine the effect of dimensionality reduction by PCA. We did PCA on the **61-LOG-cons-nqua-norm** feature set (Figure 7), and we observed that the classification accuracy was preserved, as long as we retain the first 18 components (or more).

4.2 Visualization

The features described in the Section 3.2 have dimensionality higher than three. For visualization, we have to reduce the dimensionality to three. Although it is common to use principal component analysis (PCA) [2] in order to reduce dimensionality for classification, we propose a more intuitive method.

The intuition is that different motions will exercise different body parts: for example, “walking” will have a balance between upper body and lower body, while “golf swing” will have more energy on the upper body. We propose to capture these differences with new features, namely, the ratio r of the approximate kinetic energy of groups of body parts.

First we sum up the average of the approximate kinetic energy of the joint rotation in order to

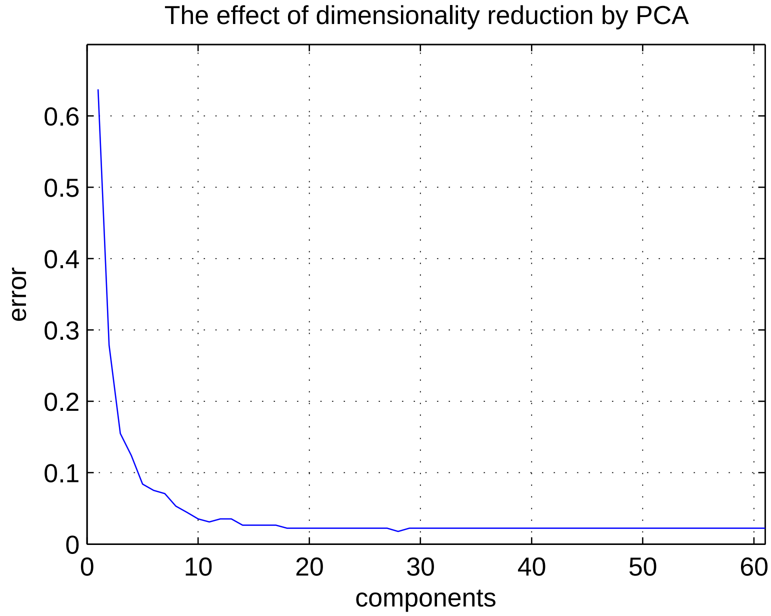


Figure 7: The error rate versus number of retained Principal Components. Notice the flattening, at about 10 components.

estimate the kinetic energy of the whole body, upper body, lower body, limbs and trunk.

$$\overline{E}_{parts} = \sum_{j \in parts} \overline{E}_j \tag{5}$$

Then we take the ratio of the kinetic energy between symmetrical body parts. As in Equation (4), we also use the $\log(x + 1)$ transform:

$$\begin{aligned} e_{all} &= \log\left(\overline{E}_{total} + 1\right) \\ r_{u/l} &= \log\left(\frac{\overline{E}_{upper} + 1}{\overline{E}_{lower} + 1}\right) \\ r_{l/t} &= \log\left(\frac{\overline{E}_{limbs} + 1}{\overline{E}_{trunk} + 1}\right) \end{aligned} \tag{6}$$

We propose to use the 3-d vector $\vec{\beta} = [e_{all}, r_{u/l}, r_{l/t}]^T$ as a feature vector of a motion capture sequence, and, again, we use the corresponding Euclidean distance between two such feature vectors β_{Ni}, β_{Mi}.

Here we show that, even with 3-d feature sets, we can still have a useful visualization. We use the features which we showed in the previous subsection (**3-LOG, 3-LIN**), as well as the first three principal components of PCA from the *61-LOG-cons-nqua-norm* feature set (**3-PCA**).

Tables 1-4 show that *3-LOG-est-nqua-norm* led to 11.50% classification error, outperforming the other *3-LOG*, *3-LIN*, and *3-PCA* features (the latter with 15.49% as shown in Figure 7). This result was another pleasant surprise: the human intuition behind *3-LOG-est-nqua-norm* won over PCA, which is mathematically optimal under the L2 norm.

Thus, we use the *3-LOG-est-nqua-norm* feature set for visualization. Figure 1 shows the scatter-plot of motion capture sequences in this 3-d space. The scatter-plot leads to observations that agree with our intuition, underlining the effectiveness of our chosen feature sets:

- The trunk energy during *walking on uneven terrain* is higher than during a normal walk.
- *Frame-climbing* requires roughly the same energy of all body parts.
- *Running* and *walking* have similar proportions of energy (upper body vs. lower body and limbs vs. trunk).

Moreover, the scatter-plot can help us spot outlier motions. For example, the points inside the red circle of Figure 1, correspond to actions with high energy; closer inspection shows that they are noisier, and the first derivatives skyrocket. The offenders correspond to *ballet dancing* motions, labeled as #5-6 and #5-8 in `http://mocap.cs.cmu.edu/`. Figure 8 shows the energy versus time for motion #5-6 (each line corresponds to a DOF). Notice that there are some large changes around frame number 150.

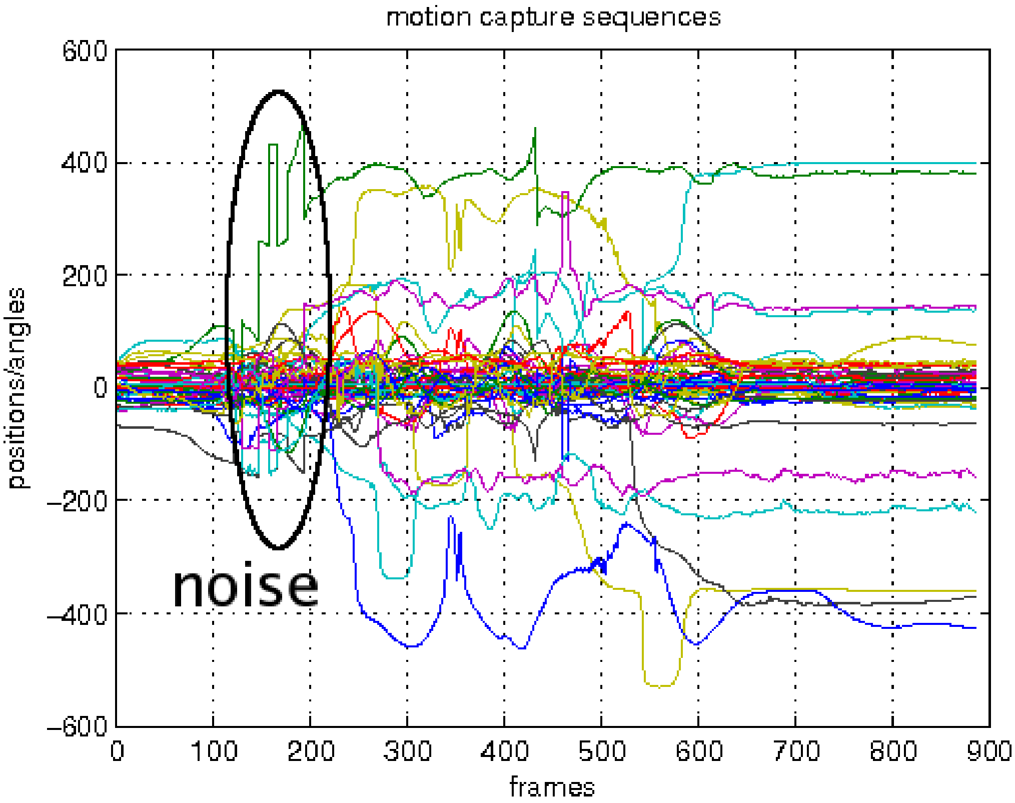


Figure 8: ’Outlier’ motion capture sequences detected by Figure 1

5 Conclusions

The goal of this paper is to find an effective and fast-to-compute distance function between two unequal-length motion capture sequences. Our main contribution is that we proposed a low-dimensionality feature-set for each motion capture sequence. After extensive experiments of 112 possible variations on a large real motion capture dataset, we propose two methods, 61-LOG-cons (most accurate) and 3-MAN (best for visualization). In all variations, the idea is to consider the total approximate kinetic energy expended by each of the approximately 70 angles in the data. The resulting feature sets achieve the original design goals:

- *Speed:* Our proposed distance function is *fast* to compute, independent of the duration of the motion capture sequences.
- *Effectiveness:* 61-LOG-cons gives good classification accuracy, and provides an excellent starting point for dimensionality reduction (with PCA, or 3-MAN), for visualization, clustering, and outlier detection (see Figure 1).

A promising direction for future work is to extend this approach to subsequence search. The distance functions we proposed in this paper are applicable only to whole sequences, but if we could also have a distance function for subsequences of motion capture data, we could apply this method to various tasks, such as finding stitching point between two motion sequences, detecting the exact frame which is noisy, and so on.

References

[1] Raffay Hamid, Amos Johnson, Samir Batta, Aaron Bobick, Charles Isbell, and Graham Coleman. Detection and explanation of anomalous activities: Representing activities as bags of event n-grams. In *Proc. of CVPR*, volume 1, pages 1031–1038, 2005.

[2] I. T. Jolliffe. *Principal Component Analysis*. Springer, 1986.

[3] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proc. of VLDB*, pages 406–417, 2002.

[4] Eamonn Keogh, Themistoklis Palpanas, Victor B. Zordan, Dimitrios Gunopulos, and Marc Cardle. Indexing large human-motion databases. In *Proc. of VLDB*, pages 780–791, 2004.

[5] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3):473–482, 2002.

[6] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. In *Proc. of SIGGRAPH*, pages 491–500, 2002.

[7] Liu Ren, Alton Patrick, Alexei A. Efros, Jessica K. Hodgins, and James M. Rehg. A data-driven approach to quantifying natural human motion. *ACM Trans. Graph.*, 24(3):1090–1097, August 2005.

[8] Arno Schödl, Richard Szeliski, David Salesin, and Irfan A. Essa. Video textures. In *Proc. of SIGGRAPH*, pages 489–498, 2000.

[9] Nikolaus F. Troje. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision*, 2(5):371–387, 2002.

[10] Victor B. Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. *ACM Trans. Graph.*, 24(3):697–701, 2005.

A Conversion of Euler angles to quaternion

Because the original motion capture data are expressed by Euler angles, it contains the problem of "gimbal lock". To avoid this problem, we calculate a set of anglar velocities by converting Euler angles to quaternions. If the original motion capture data are described as XYZ system of Euler angles, the corresponding quaternion can be obtained as:

$$\mathbf{q} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) & + & \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) & - & \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) & + & \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) & - & \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}$$

where **q** indicates a quaternion. And the relation between the quaternion and angular velocity is defined as:

$$\frac{d\mathbf{q}}{dt} = \frac{1}{2}\omega(t) \otimes \mathbf{q} \tag{7}$$

where $\omega(t) = [0\ \omega_x(t)\ \omega_y(t)\ \omega_z(t)]^T$ indicates an angular velocity vector represented as a quaternion and $\otimes$ denotes quaternion multiplication. The quaternion multiplication is defined as:

$$\hat{\mathbf{q}} \otimes \mathbf{q} = \begin{bmatrix} -q_1 & -q_2 & -q_3 & q_0 \\ q_0 & q_3 & -q_2 & q_1 \\ -q_3 & q_0 & q_1 & q_2 \\ q_2 & -q_1 & q_0 & q_3 \end{bmatrix} \begin{bmatrix} \hat{q_0} \\ \hat{q_1} \\ \hat{q_2} \\ \hat{q_3} \end{bmatrix}$$

where $\mathbf{q} = [q_0\ q_1\ q_2\ q_3]^T$ and $\hat{\mathbf{q}} = [\hat{q_0}\ \hat{q_1}\ \hat{q_2}\ \hat{q_3}]^T$.

We can obtain the estimate of angular velocity($\omega(t)$) by solving the formula of (7), where $d\mathbf{q}/dt$ and **q** are calculated from the original motion capture data by converting to quaternions. Notice that the formula of (7) is overdetermined because it has four equations and the number of unknown variables are three $(\omega_x(t), \omega_y(t), \omega_z(t))$. So we calculate $\omega(t)$ using least squares.