Movement Classes from Human Motion Data

Kang Hoon Lee[1], Jong Pil Park[2], and Jehee Lee[2]

[1] Kwangwoon University, Seoul 139-701, Korea
kang@kw.ac.kr
[2] Seoul National University, Seoul 151-742, Korea
{jppark,jehee}@mrl.snu.ac.kr

Abstract. We present a new method for identifying a set of movement types from unlabelled human motion data. One typical approach first segments input motion into a series of intervals, and then clusters those into a set of groups. Unfortunately, the dependency between segmentation and clustering causes trouble in alternate tuning of parameters. Instead, we unify those two tasks in a single optimization framework that searches for the optimal segmentation maximizing the quality of clustering. The genetic algorithm is employed to address this combinatorial problem with our own genetic representation and fitness function. As the primary benefit, the user is able to obtain a repertoir of major movements just by selecting the number of classses to be identified. We demonstrate the usefulness of our approach by providing visual descriptions of motion data, and an intuitive animation authoring interface based on movement collections.

Keywords: computer animation, human motion data, movement classfication.

1 Introduction

Captured human motion data is widely used today in various applications such as game development, film production, and sports analysis. Given a large amount of motion data for practical use, it is often challenging for the user to quickly recognize what kinds of motions are available, or whether some specific kinds of motions need to be additionally acquired. For example, let us assume that we have a motion clip in which hundreds of dance steps are included. Manually examining the entire sequence not only is cumbersome, but also hardly provides a complete view of major dance steps, such as 'box step', 'free spin', and so on.

Many researchers have addressed this problem by presenting automatic methods of identifying a set of movement types from unlabelled motion data. A typical approach first partitions input motion data temporally into a collection of motion segments, and then clusters those into several groups such that similar segments can be included in the same group. The result of this two-phase approach strongly depends on the first phase, motion segmentation. As an extreme case, if we use a uniform segmentation scheme of partitioning input motion at regular intervals, it is hardly expected for any clustering algorithms to produce

Z. Pan et al. (Eds.): Transactions on Edutainment VI, LNCS 6758, pp. 122–131, 2011.

meaningful categories. In fact, there are a lot of more elaborate segmentation schemes than uniform partitioning, and suitable combinations of segmentation and clustering methods can yield much better results. However, there is not yet a consensus about any specific combinations that guarantee reasonable quality without respect to the domain of input motion.

We present an alternative approach to automatic motion classification from unlabelled human motion data. Instead of regarding segmentation and clustering as two separate procedures, we unify those into a single optimization framework in which both the segmentation of input motion and the clustering of segmented intervals can be simultaneously achieved. Given only the number of classes to be identified, we search for an optimal segmentation that produces the most concise and precise collection of movement classes. In order to cope with a large diversity of lengths and arrangements of basic movements, we allow both the lengths and the number of segments to vary within a pre-defined ranges through our optimization process. As a result, hundreds of irregular size segments as well as dozens of regular size segments can be equally treated as candidate segmentations for the same motion. Such a few constraints give us a huge size of search space for optimization, which could not be practically handled by exhaustive search.

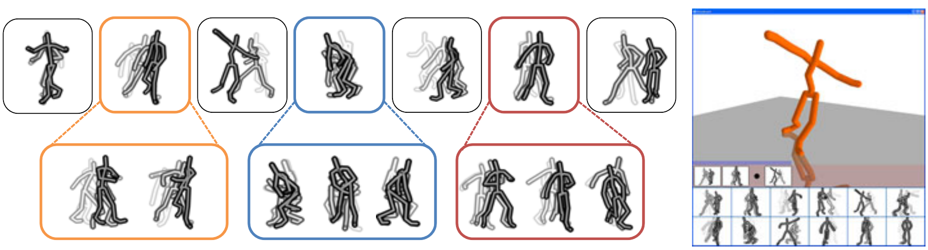

Fig. 1. Movement classes from breakdance motion data of about two thoudsands of frames. (Left top) The centroid motion segment of each class. (Left bottom) Some example motion segments other than the centroids. (Right) Our interactive animation authoring interface based on classified movements.

We employ the genetic algorithm to find a good approximate of the global optimum from such a large combinatorial space. To do so, we devise a genetic representation of the solution domain so that any feasible segmentations can be compactly encoded as binary strings (see Section 3). The fitness of a string, in other words, the goodness of a segmentation, is evaluated according to the quality of the clusters formed by running the *K-medoids* algorithm on the segmented intervals. Our measure of clustering quality prefers more compressed, and less lossy classifications (see Section 4). In optimization, a population of initial strings are randomly generated, which iteratively evolve into the next generations through selection, crossover, and mutation (see Section 5). When the optimization terminates, the string of the highest fitness is regarded as the best segmentation. The clusters formed by running the *K-medoids* finally on the best segmentation corresponds to our collection of movement classes, each of which is best represented by the centroid segment of its cluster (see Figure 1 (Left)).

The main benefit of our method is that the user can acquire a precise repertoire of major movements easily and consistently. The user provides only the number of motion classes without caring about which segmentation scheme is appropriate and how finely or coarsely input motion should be partitioned. We demonstrate the flexibility and usefulness of our approach by summarizing a variety of input motion data visually with a concise set of still images of classified movements (see Figure 2). Furthermore, our experimental interface system shows that the user can intuitively author new animation sequences just by arranging movement classes at any intended orders (see Figure 1 (Right)).

2 Background

The motion capture technology opened a new way of expressing highly realistic motion within virtual worlds for computer games and movies. In order to express a large diversity of motions using a limited amount of motion data, a lot of methods for processing motion data have been investigated. Currently, we are able to edit a motion sequence to satisfy geometric constraints [1], blend multiple sequences into intermediate motions [2], rearrange sub-sequences into arbitrarily long sequences that satisfy high-level constraints [3,4,5,6].

Data-driven motion synthesis often requires input motion data be preprocessed into computationally efficient and easily recognizable structures. Identifying movement types is an important step for compact representation of original data. Clustering after segmentation has been a popular approach to address this problem, using various segmentation schemes. Barbič et al. employed the principal component analysis (PCA) to find cutting locations where intrinsic dimensionality is changed [7]. Fod et al. identified zero-crossing frames, where angular velocities of joints reverse, as segmenting positions [8]. Kwon and Shin decided cutting positions based on the vertical peaks of the center-of-mass trajectory [9]. These methods regard segmentation and clustering as two separate tasks, while we simultaneously perform both tasks in a single unified procedure.

Our problem is closely related to another problem of retrieving a set of similar motions from motion database given a query motion. Typically, this involves the measure of evaluating the similarity of any two motion sequences, which has been addressed by several different approaches based on joint angles [3], point clouds [4], geometric relational characteristics [10]. Efficient methods for motion retrieval has been explored based on those measures. Kovar et al. presented a match web structure for searching over a large motion database to identify a set of similar motions efficiently [11]. Meng et al. pre-computed the best matches for each frame in a compact structure that supports fast motion retrieval [12].

Recently, there have been efforts for handling segmentation and clustering in a coupled manner as in ours. Beaudoin et al. presented a method of finding repetitive motion patterns, called motion-motifs [13]. This method is highly scalable to the size of input motion, and produces a well-organized view of the entire motion data. However, its randomized approach does not guarantee clear distinction among identified motifs, which is the goal of this paper. The most similar approach to ours is found in the work of Zhou et al. [14], where they also formulate the segmentation problem as an optimization of clustering. The key difference from ours is that the number of segments is constant, and regarded as the input parameter in Zhou et al. On the other hand, an arbitrary number of segments are allowed in our segmentation, which leaves only the number of classes as the input parameter, and gives much higher flexibility.

3 Genetic Representation

Let us denote the number of classes and the maximum number of segments as K and N, respectively. Then, the number of segments n is allowed to have an arbitrary value within the range of $[K, N]$ in our method. The lower boundary condition $n \geq K$ guarantees that every cluster includes at least a single segment. On the other hand, the upper boundary condition is to limit the size of search space, which grows exponentially according to N.

We further reduce the search space by pre-allocating $N - 1$ possible cut locations within the range of $(1, T)$ where T is the number of motion frames. For example, let the maximum number of segments N be two. Then, just a single location, e.g. $T/2$, is pre-allocated, where a cut can be inserted or not. If inserted, we obtain two disjoint segments $[1, T/2)$ and $[T/2, T]$. Otherwise, only a single segment is obtained, corresponding to the entire interval $[1, T]$.

This simplification allows us to rephrase our problem as determining whether to accept or not each element from the ordered set of every possible cut location $\mathbf{T} = \{T_1, T_2, \cdots, T_{N-1}\}$. Denoting acceptance by 1 and rejection by 0 allows us to encode any segmentations in the solution domain compactly as binary strings $\mathbf{b} = b_1 b_2 b_3 \cdots b_{N-1}$, where $b_m \in \{0, 1\}$. Note that at least $K - 1$ values of $\mathbf{b}$ should be 1 to satisfy the lower boundary condition $n \geq K$ as mentioned above.

In our experiments, we built the set of possible cut locations $\mathbf{T}$ simply by sampling the entire interval regularly (e.g. every half or one second). Any other more elaborate schemes, such as sampling where the contact states of feet are changed, might also be employed in deciding $\mathbf{T}$ to obtain more meaningful and smaller search spaces. However, manually selecting and tuning a specific scheme for each kind of input motion is tedious and error-prone, which reduces the main benefit of our approach, that is, removing the need of such a special attention in segmetnation. Our experimental results show that the brute-force regular sampling yields satisfying quality for a diversity of input motions.

4 Fitness Evaluation

In order to evaluate the fitness of a string $\mathbf{b}$, we first decode it into a series of cutting locations $\mathbf{t} = \{t_1, t_2, \cdots, t_{n-1}\} \subset \mathbf{T}$ by including only the m-th elements of $\mathbf{T}$ whenever $b_m = 1$. Those cutting locations are easily interpreted as a series of segment intervals, which are then clustered into K groups according to their similarity. Finally, the quality of clustering is measured as the score of $\mathbf{b}$ based on our fitness function. This section explains three orthogonal components involved in this process in detail: motion similarity measure, clustering algorithm, and clustering quality measure.

4.1 Motion Distance

Let the input motion data of our method be a sequence of motion frames $\{\mathbf{x}_t | 1 \leq t \leq T\}$ where $\mathbf{x}_t$ denotes a skeletal pose at an instance. Each pose is described as a root position $\mathbf{p}_t$ and a set of joint orientations $\{\mathbf{q}_{t,j} | 1 \leq j \leq J\}$. Given any two frames $\mathbf{x}_t$ and $\mathbf{x}_{t'}$, we evaluate the dissimilarity $\mathrm{d}(\mathbf{x}_t, \mathbf{x}_{t'})$ between the corresponding poses by using the measure of Lee et al. [3].

This pose-to-pose distance measure can be easily extended to evaluate the distance between any two motion intervals $[t_s, t_e]$ and $[t'_s, t'_e]$. Because any two target intervals can have different lengths, temporal alignment of those intervals is a prerequisite for pose-to-pose comparison. The dynamic time warping is a well-known approach to this problem ensuring optimal alignment, but its computational cost is relatively expensive for our purpose.

Instead, we use a simple uniform time warping such that the m-th frame of the longer interval corresponds to the $\lfloor ml'/l \rfloor$-th frame of the shorter one, where l and l' are the lengths of the longer and shorter intervals, respectively. If the matched pairs of frames after alignment are $\{(t_m, t'_m) | 1 \leq m \leq L = max(l, l'), t_s \leq t_m \leq t_e, t'_s \leq t'_m \leq t'_e\}$, then our motion distance is evaluated as follows.

$$\mathrm{D}([t_s, t_e], [t'_s, t'_e]) = \frac{1}{L} \sum_{m=1}^{L} \mathrm{d}(\mathbf{x}_{t_m}, \mathbf{x}_{t'_m}) \tag{1}$$

This calculation is one of the most frequently used operations in our optimization process, so reducing its computational cost can significantly increase the overall performance. For this purpose, we pre-compute $\mathrm{d}(\mathbf{x}_t, \mathbf{x}_{t'})$ for every pair of (t, t') and just look up the resulting distance table at the optimization phase.

4.2 Motion Clustering

The cutting locations $\mathbf{t} = \{t_1, t_2, \cdots, t_{n-1}\}$ are interpreted as n segment intervals $\{\mathbf{s}_1 = [1, t_1), \mathbf{s}_2 = [t_1, t_2), \cdots, \mathbf{s}_n = [t_{n-1}, T]\}$. We use the *K-medoids* algorithm to parition those segment intervals into a set of clusters $\mathbf{C} = \{\mathbf{C}_k | k = 1, \cdots, K\}$. We define each cluster $\mathbf{C}_k$ as a collection of segments $\{\mathbf{s}^i_k | 1 \leq i \leq N_k\}$ where N_k is the number of segments included in the cluster.

The overall process of the *K-medoids* algorithm is almost equivalent to the *K-means* algorithm except that the *K-medoids* algorithm allows more general distance measures other than the Euclidean distance. The objective function of the *K-medoids* is defined as follows.

$$\mathrm{E}(\mathbf{C}) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \mathrm{D}(\mathbf{s}_k^i, \mathbf{s}_k^c) \tag{2}$$

where $\mathbf{s}_k^c$ and $\mathbf{s}_k^i$ represent the central segment (i.e. centroid) and the i-th segment belonging to the k-th cluster, respectively, and $\mathrm{D}(\mathbf{s}, \mathbf{s}')$ corresponds to $\mathrm{D}([t_s, t_e], [t'_s, t'_e])$ in Equation 1 when $\mathbf{s} = [t_s, t_e]$ and $\mathbf{s}' = [t'_s, t'_e]$.

In order to minimize this objective function, we first initialize the centroids of K clusters as the first K segments $\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_K$, and then iteratively update clusters and their centroids until convergence. When updating clusters, we associate every segment with its closest cluster based on the distances to centroids. For each new cluster, its centroid is updated to a new one that has the minimal sum of distances to every other segment within the same cluster. We terminate this iteration when no change of segment-to-cluster association is found.

4.3 Cluster Evaluation

We evaluate the result of clustering $\mathbf{C}$ via two contradictory features, compression ratio and reconstruction error, to achieve a reasonable trade-off between the conciseness and the preciseness of the action classes. The compression ratio measures how short segments are required to summarize the input motion data in comparison with the entire length as follows, denoting the length of a segment $\mathbf{s}$ as $L(\mathbf{s})$.

$$\mathrm{R}(\mathbf{C}) = \frac{1}{T} \sum_{k=1}^{K} L(\mathbf{s}_k^c) \tag{3}$$

On the other hand, the reconstruction error measures how much distortion is produced when we summarize the original motion only with the identified actions. This roughly corresponds to the data loss after replacing every segment $\mathbf{s}_k^i$ with the centroid segment $\mathbf{s}_k^c$ for every cluster in the input motion data. We evaluate this error by slightly modifying the function $\mathrm{E}(\mathbf{C})$ as follows in order to limit its range to $[0, 1]$ without respect to the length of input motion and the diversity of poses, where d_{max} is the maximum among the distance of every pair of two poses $\mathrm{d}(\mathbf{x}_t, \mathbf{x}_{t'})$.

$$\mathrm{E}'(\mathbf{C}) = \frac{1}{\mathrm{d}_{max}} \sum_{k=1}^{K} \sum_{i=1}^{N_k} \frac{L(\mathbf{s}_k^i)}{T} \mathrm{D}(\mathbf{s}_k^i, \mathbf{s}_k^c) \tag{4}$$

Choosing only one of these two features might cause undesirable effects in our optimization. Simply igonoring the reconstruction error, low compression ratio means just a large number of short segments that are grouped into arbitrary action classes without regard to thier similiarities. On the other side, low reconstruction error corresponds to a small number of long segments. At its extreme, the lowest error can be easily achieved by segmenting the input motion into K intervals, and assigning a single segment for each cluster.

We take the inverse of the weighted sum of both features as our fitness function as follows in order to generate higher score for lower sum of compression ratio and reconstruction error, where the weight ω controls the relative contributions of the two features to the final fitness ($0 \leq \omega \leq 1$).

$$\mathrm{F}(\mathbf{C}) = \frac{1}{\omega \mathrm{R}(\mathbf{C}) + (1-\omega)\mathrm{E}'(\mathbf{C})} \tag{5}$$

5 Optimizing Classifications

We are now ready to perform the genetic algorithm with our genetic representation of the solution domain and the fitness function for any given solutions. A set of initial solutions, called a population, is randomly generated first. Then, the population is iteratively reproduced into a new population in the next generation based on three basic rules: selection, crossover, and mutation. This iteration terminates when the population converges to a local optimum, or the number of generations reaches to a pre-defined threshold. Finally, the best string in the last generation is selected as our classfication.

The initial population $\{\mathbf{b}_p | p = 1, \cdots, P\}$ is randomly generated by selecting each value of the string $\mathbf{b}_p$ from $\{0, 1\}$ with an equal probability. Then, we need to check if the number of 1's in $\mathbf{b}_p$ is greater than or equal to $K-1$ for every string, as discussed in Section 3. If not, we locally update some 0's to 1's to obtain a valid string. This validation process applies not only to the initialization phase, but also to every reproduction step.

We adopt *elitism* and *roulette-wheel selection* schemes when producing the next generation. The score for every string $\{\mathbf{b}_p\}$ is evaluated by our fitness function as discussed in Section 4. A small portion of the entire population that obtains the topmost score is regarded as the *elite*, which is transferred to the next generation in its original form. For the rest of the population, we associate each string with its reproduction probability as the ratio of its own score to the total score. Then, each new string in the next generation, e.g. a child, is produced by probabilistically selecting a pair of existing strings, e.g. parents, and performing crossover and mutation operations.

Among many crossover techniques, we use the one-point crossover that cuts two existing strings $\mathbf{b} = b_1 \cdots b_{N-1}$ and $\mathbf{d} = d_1 \cdots d_{N-1}$ at a common location p, and exhanges the resulting substrings with one another to reassemble two new strings $\mathbf{b}' = b_1 \cdots b_{p-1} b_p d_{p+1} \cdots d_{N-1}$ and $\mathbf{d}' = d_1 \cdots d_{p-1} d_p b_{p+1} \cdots b_{N-1}$. Both new strings are accepted into the population of the next generation in our implementation. When deciding cut locations, we prefer the locations where both values b_p and d_p are 1 to preserve existing segment intervals. If there are no such locations, we randomly select one among the locations where either b_p or d_p is 1.

We probabilistically mutate each new string with the probability being decreased through iterations. In other words, a large portion of the population mutates at the early phase of our optimization for extensively exploring the

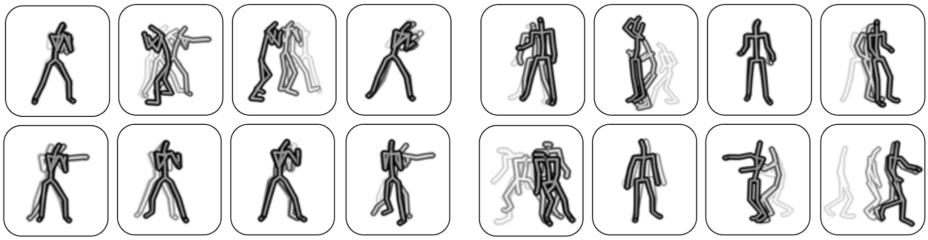

Fig. 2. Movement classes from boxing (Left) and basketball (Right) motion data of about three and five thousands frames, respectively

search space, while little portion is mutated at the late phase for finely tuning the existing solutions. In our implementation, the mutation probability is determined as $e^{-5g/G}$ where g and G is the current and the total number of generations, respectively. Once a string is determined to be mutated according to the probability, we randomly choose a single location in the string, and negate its value.

6 Experimental Results

We experimented with a variety of input motions including boxing, breakdance, and basketball. The number of classes for each motion data was determined by running optimizations with various numbers between 5 and 20, and selecting the best one yielding the highest score. In optimization, we used the same parameters for every test data. We represented a segmentation as a string by arranging possible cut locations at every one second. The population consisted of one hundred strings, which reproduced for 500 generations. For each generation, the topmost 10 strings were selected as elites, and transferred to the next generation without crossover. The usefulness of our approach is qualitatively demonstrated by visually summarizing each movement class in Figure 2. We rendered a small number of poses belonging to each segment in a cartoon style, and layered the rendered poses in sequence of time. Additionally, we utilized the same visualization techniques in creating an interface system for animation authoring (see Figure 1 (Right)). Regarding the movement classes as animation building blocks, the user can intuitively design new animation sequences by arranging movement classes in the time domain.

In order to demonstrate the effectiveness and robustness of our approach, we plot the maximum scores obtained from a short basketball motion clip of 500 frames with respect to the number of generations g on the left and the number of classes K on the right. For the left graph, we fixed K to 13, sampled possible cut locations at every 10 frame, and produced 200 generations. The resulting graph shows that the maximum score sharply increases at irregular intervales due to our policy of elitism, and eventually converges to a local optimum around the 100th generation. For the right graph, we varied K from 1 to 50, and run our optimization process 5 times for each K to cope with the randomness of the genetic algorithm. When K is 1, every segment belongs to the same cluster, leading to the lowest compression ratio and the highest reconstruction error. On the other hand, when K equals to the number of maximum segments (50), every cluster includes a single segment, thus yielding the highest compression ratio (1), e.g. no compression, and the lowest reconstruction error (0), e.g. no error. The score is equal to 4.0 because the weight ω is 0.25 in this experiment. Between 1 and 50, the scores roughly draw an arc, whose maximum occurs at the ideal number of classes (13 in this case) with respect to our fitness measure.

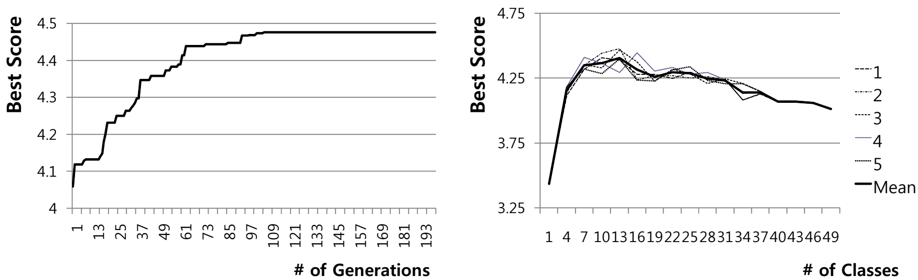

Fig. 3. The best scores obtained from our genetic algorithm with respect to the increasing numbers of classes

7 Discussion

We have presented a new method for identifying a user-given number of motion classes from an unlabelled human motion data. The key characteristic of our method is in its unified process of obtaining the optimal segmentation that produces a high quality classification. Our experimental results demonstrate that our method based on the genetic algorithm is highly flexible to a large diversity of human motions without exhaustive tuning of parameters.

The primary advantage of our method is that the user can easily obtain a simplified view of his/her motion data that summarizes the entire motion into a small set of representative movements. Furthermore, such movement collections can be utilized for interactively authoring new animation sequences. We plan to improve our authoing system introduced in this paper to incorporate path editing, pose editing, and multi-character synchronization.

Our solution domain allowing an arbitrary number of segments causes a challenging problem of combinatorial explosion, which makes it hard for our method to scale well with the size of input motion data. As an idea of dealing with this problem, we are interested in a hierarchical extension that first processes a set of short motion sequences into a large number of classes, and then merge those classes into a small number of more comprehensive classes.

Acknowledgments. We would like to thank Sang Won Lee for his help in video editing. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2010-0008243 and No. 2010-0012759).

References

1. Lee, J., Shin, S.Y.: A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures. In: 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 39–48. ACM Press, New York (1999)
2. Kovar, L., Gleicher, M.: Flexible Automatic Motion Blending with Registration Curves. In: 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 214–224. Eurographics Association, Switzerland (2003)
3. Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K., Pollard, N.S.: Interactive Control of Avatars Animated with Human Motion Data. ACM Trans. Graph. 21(3), 491–500 (2002)
4. Kovar, L., Gleicher, M., Pighin, F.: Motion Graphs. ACM Trans. Graph. 21(3), 473–482 (2002)
5. Arikan, O., Forsyth, D.A., O'Brien, J.F.: Motion Synthesis from Annotations. ACM Trans. Graph. 22(3), 402–408 (2003)
6. Treuille, A., Lee, Y., Popović, Z.: Near-Optimal Character Animation with Continuous Control. ACM Trans. Graph. 26(3), 7 (2007)
7. Barbič, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J.K., Pollard, N.S.: Segmenting Motion Capture Data into Distinct Behaviors. In: Graphics Interface, pp. 185–194. Canadian Human-Computer Communications Society, Ontario (2004)
8. Fod, A., Mataric, M.J., Jenkins, O.C.: Automated Derivation of Primitives for Movement Classfication. Auton. Robots 12(1), 39–54 (2002)
9. Kwon, T., Shin, S.Y.: Motion Modeling for On-Line Locomotion Synthesis. In: 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 29–38. ACM Press, New York (2005)
10. Müller, M., Röder, T., Clausen, M.: Efficient Content-Based Retrieval of Motion Capture Data. ACM Trans. Graph. 24(3), 677–685 (2005)
11. Kovar, L., Gleicher, M.: Automated Extraction and Parameterization of Motions in Large Data Sets. ACM Trans. Graph. 23(3), 559–568 (2004)
12. Meng, J., Yuan, J., Hans, M., Wu, Y.: Mining Motifs from Human Motion. In: Eurographics 2008–Short Papers, pp. 71–74 (2008)
13. Beaudoin, P., Coros, S., van de Panne, M., Poulin, P.: Motion-Motif Graphs. In: 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 117–126. Eurographics Association, Switzerland (2008)
14. Zhou, F., de la Torre, F., Hodgins, J.K.: Aligned Cluster Analysis for Temporal Segmentation of Human Motion. In: IEEE International Conference on Automatic Face and Gesture Recognition (2008)