芸術科学会論文誌 Vol. 9, No. 3, pp. 130 – 139

A Simple and Efficient Method to Extract Keyposes from Mocap Data

Sindharta Tanuwijaya **Yoshio Ohno**

Keio University
{sin, ohno} @ on.ics.keio.ac.jp

Abstract
Researchers have been constantly developing new methods for generating realistic human motion. A common and popular approach is to use motion capture data or mocap data. However, this approach produces a mountain of data and there is an increasing necessity for quick and efficient mocap data search methods.

For text documents, search results are usually returned in the form of a list of documents' titles. This allows users to quickly browse the search results and easily find the documents that they are looking for. Applying this method to mocap data requires the extraction of keyposes since displaying every frame will only present excessive and unnecessary information.

The authors of this paper propose a simple and efficient method to automatically extract keyposes of mocap data. The proposed method requires only $O(kn)$ operations. Thus, users can vary the search parameters and results will be presented interactively. Our results show that the proposed approach is applicable to mocap data of complex motions and motions composing of many motion segments.

Keyword
mocap, data, keypose, search, retrieval, animation

1 Introduction

Motion capture is a popular technique used in creating smooth animations of a human performing certain actions. The data is taken by periodically sampling the movement of the sensors, which are attached to the performer's body, and then cleaned up by removing defects and smoothing measurement errors or noises that might have occurred during the sampling process. These defects may be caused by the limitation of the hardware itself, for example: the occlusion of optical sensors in optical motion capture devices causes the position of the sensors to be undetected for a period of time. For the purpose of simplicity, the data, which usually comes initially in the form of joint positions in each frame, may be converted into another representation, for example a skeleton hierarchy of joint angles.

Although motion capture data, commonly called mocap data, was not commonly used outside entertainment genres such as games and movies, partly due to the bulkiness and high cost of the required devices, it can be expected that mocap data will become commonplace in the near future as cheaper and more practical devices are developed [14].

Initially, mocap data retrieval or search received little attention due to a lack of or relatively few mocap data. However, a large amount of mocap data has been accumulated and made accessible since then, for example: [3]. It is this accumulation of data that has lead to an increasing demand for mocap data search.

However, the research of displaying the results of mocap data retrieval has just begun to appear, for example: the research in [2] provides a method to make a synopsis of mocap data by automatically selecting the keyposes of the data and rendering them into an image, so that the user can understand the motion without actually watching the whole animation. Certainly, watching the animation of every motion in order to find the exact motion that the user wants is time and energy consuming. Furthermore, representing mocap data as images enables those mocap data to be displayed as a list, as can be seen in [5].

This paper focuses on making a synopsis of a mocap data by automatically determining its keyposes, representing the motion as a whole. Our approach can be executed very efficiently and effectively, allowing users to input the number of keyposes or joint weights, and see the results immediately. This efficiency becomes even more essential if we want to create a mocap database on the Internet, and allow users to upload their own mocap data to our system. Approaches with heavy computation is not practical because they will put much burden on the system if a lot of mocap data are simultaneously uploaded.

2 Related Work

The problem of mocap data retrieval has been addressed as a preliminary step before synthesizing new motions as described in [1], and [9]. In general, the methods used to search mocap data can be divided into two categories. The first category is by giving annotations to each frame based on a classifier learned by using training data, so that ordinary textual search can be performed. The second category is by using content-based retrieval or query by example, in which a motion query is initially given as a basis to locate other motions in the database that are similar to the motion query [4].

In most research, joint positions, angles or their derivations are used as inputs to the systems. Since logically similar motions need not be numerically similar as pointed out by [9], these systems may fail to return all logically similar motions as search results. To address this problem, [11] proposed using geometric features that describe the relationship among joint positions as inputs instead of their positions or angles.

The problem of extracting keyposes was first addressed by [2], where the identification of keyposes is performed by embedding the motion in a low dimensional space and analyzing it. This method uses a replicated multidimensional scaling (RMDS) algorithm to reduce the dimensions of the data and then identifies extremum points by using an iterative algorithm. This method is then extended in [7] to accommodate their use of feature values from [11]. In contrast, [8] modeled the keypose extraction problem as a constrained matrix factorization problem and used an optimization technique to solve it, while [10] used an optimization technique to find a set of poses, in which the total distances of neighboring keyposes are maximized.

Our approach, although similar to [10], differs in that we focus on obtaining representative keyposes which are distinct, even if the motion is cyclic, e.g a running motion. The method proposed in this paper can be effectively used in such situations as described in [13].

To the data mining community, data such as mocap data belongs to a category called time series data, the research of which has been well documented. However, the majority of the research has focused on one dimensional continuous data. For example: [6] and [12] presented techniques to detect significant points in one dimensional time series data. It is possible then, to think of applying the same method to mocap data and approximating the significant points to be keyposes of the motion. Unfortunately, applying these techniques to

mocap data, which is inherently a multidimensional time series data, proves to be much harder, if not impossible.

3 Approach

The input to our approach is a single motion capture data with any arbitrary length, or in other words all the frames inside a single motion capture data M, i.e $M = [f_1, f_2, \ldots, f_n]$ where n refers to the number of frames in the motion data. Unlike [1] which uses time windows or groups one frame with its neighboring frames, we treat a single frame as one single data which is unrelated to all other frames.

Despite treating a single frame as unrelated to its neighbors, mocap data actually represents a continuous motion, which means that the distances of the joint positions or angles in one particular frame to the joint positions or angles in its neighbor frames are small. This implies that when one particular frame has been chosen as a keypose, it is very straightforward to give penalties to its neighboring frames by taking their distances into account, so that they will less likely be chosen as keyposes at the next iteration.

In calculating the distance between frames, we take only the joint positions into account forming a vector of $V = [j_1^x, j_1^y, j_1^z, \ldots, j_m^x, j_m^y, j_m^z]$ where m means the number of joints. We refrain from using joint angles because of the nature of rotation which is not Euclidean, making the computation of the distance of rotations relatively expensive compared to the computation of the distance of positions. In addition, from our experiment results, it can be seen that using joint positions is adequate for our purpose.

In the next subsection, we will introduce the concept of our approach to extract keyposes from mocap data. Then we will present an optimized algorithm based on the concept, in which the complexity grows linearly with the number of keyposes and frames. We end this section by comparing our approach to an optimization approach using a simple dataset.

3.1 Extracting Keyposes

Initially, we assume k as the number of keyposes that a user wants to find in a mocap data. Then we calculate the mean of the joint positions of all frames in the mocap data to be used as a reference point.

After the preliminary steps are completed, we define the frame which has the longest distance to the reference point, i.e., the mean of the joint positions, as the first keypose. Next, for each frame, or keypose candidate, we calculate the distances to its nearest keypose and to the reference point, and assign the shorter distance to the frame. Then, we pick the frame which has the highest assigned value as the second keypose. We use Euclidean distance to measure these distances and repeat these steps until the number of keyposes reaches k. The illustration of these steps can be seen in Figure 1.

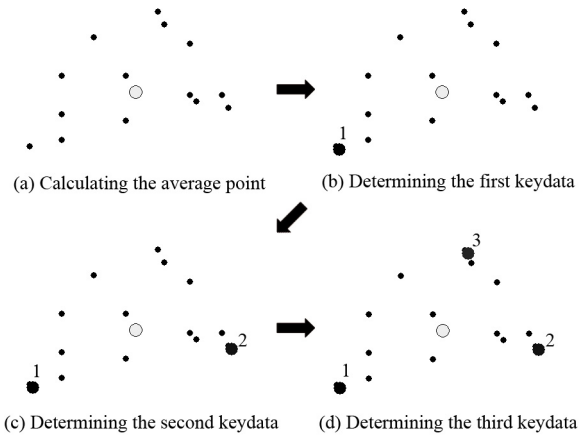


Fig. 1: Extracting Keyposes. Unfilled large circle denotes the reference point, small circles denote the data, while filled large circles denote the keyposes.

These selected frames basically represent extreme body poses which have the longest distances from other body poses in the motion, and therefore we are convinced that they are proper candidates for keyposes of a motion. When users look at such multiple extreme poses at the same time, it is relatively easy to visualize the poses inbetween although they are actually not displayed. Conversely, it is difficult, if not impossible, to visualize the extreme body poses when only the inbetween poses are shown to users. These two reasons convince us that our simple and efficient approach explained above can extract keyposes from mocap data.

Since our approach is very efficient, the problem of determining the best value of k or the number of keyposes can be left to the user to decide, while still getting the results in real time. Naturally, long motions will require high values of k, while low values will be adequate for short motions.

3.2 Optimized Algorithm

If the number of frames in a mocap data is n, the number of joints considered in the calculation is m and the required number of important poses is k, then the cost required to determine the keyposes using the optimized algorithm is $O(kmn)$, in which k is generally very low. If the value of m is also low, then we can approximate the complexity to be $O(kn)$. Using the above notations, the optimized algorithm is defined in Algorithm 1.

Input: The joint positions of all the frames in a motion capture data M and the number of keyposes k
Output: The keyposes K
$K \leftarrow [\,]$;
$R \leftarrow$ CalculateRefPoint(M);
$i \leftarrow 0$;
while $i < n$ **do**
  $D_i \leftarrow$ Distance(f_i,R);
  assign D_i to f_i;
  increment i;
end
while *the number of elements in* $K < k$ **do**
  $f \leftarrow$ the frame which has the longest D_i;
  add f into K;
  $i \leftarrow 0$;
  while $i < n$ **do**
    $d \leftarrow$ Distance(f_i,f);
    if $d < D_i$ **then**
      $D_i \leftarrow d$;
    end
    increment i;
  end
end
return K;

Algorithm 1: The optimized algorithm to extract keyposes from mocap

In contrast, the complexity of the approach in [2] is $O(kn^2)$ for the preliminary steps to compute the distances between frames, which is then added to the complexity of solving an optimization problem to reduce the number of dimensions. These steps should be done before the approach is able to extract the keyposes iteratively. One may consider trying this approach without reducing the number of dimensions to reduce the complexity of the process. However, this dimensionality reduction is crucial in this approach as stated in the paper, and without it, we show that our results are better in Figure 2.

3.3 Comparison

Since high dimensional data cannot be easily visualized, we use a simple one dimensional sine wave dataset to illustrate the differences of our approach to the optimization technique described in [10] for the sake of clarity. As can be seen in Figure 3, this sine wave dataset is similar to mocap data in that it is continuous, it has both local and global maximum (minimum) points, and it may or may not be periodical depending on the interval.

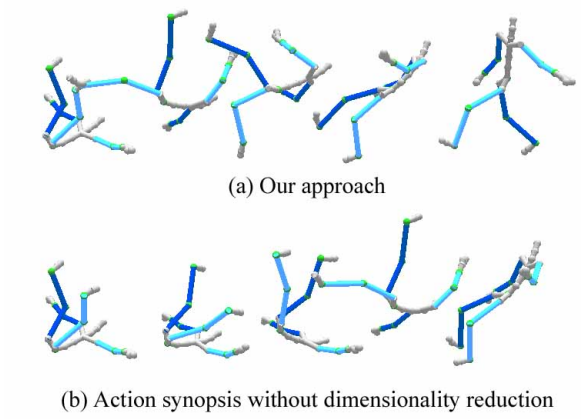


Fig. 2: A comparison of our approach and [2] without dimensionality reduction.

For the optimization technique, we use a maximization function $\Sigma_i^{k-1} D_{f_i,f_{i+1}}$ with a constraint $t(f_i) < t(f_{i+1})$ where k denotes the number of keydata, f_i denotes the i-th keydata, and $t(f_i)$ denotes the time when f_i appears. In the figure, we can see that although [10] also obtains representative keydata, some of them are the same as other keydata, i.e. they have the same values. Meanwhile, our method obtains representative keydata, the values of which are distinct. We further this comparison by applying both approaches to mocap data in the next section.

At the same time, note that our approach manages to obtain keydata consistently, i.e. the keydata obtained by lower k values still exist when we use higher k values, unlike the results of [10], in which the keydata located second from the left for $k = 5$ is not chosen again as a keydata for $k = 7$. Further, it is straightforward for our approach to extract only one keypose, i.e $k = 1$.

4 Experiment

Most of our data come in the form of a skeletal hierarchy of Euler joint angles, with the coordinate system shown in Figure 4. We convert this representation into three dimensional joint positions by ignoring global X and Z translations of the root joint because we choose to regard XZ planar transformations of the root joint as irrelevant in calcu-

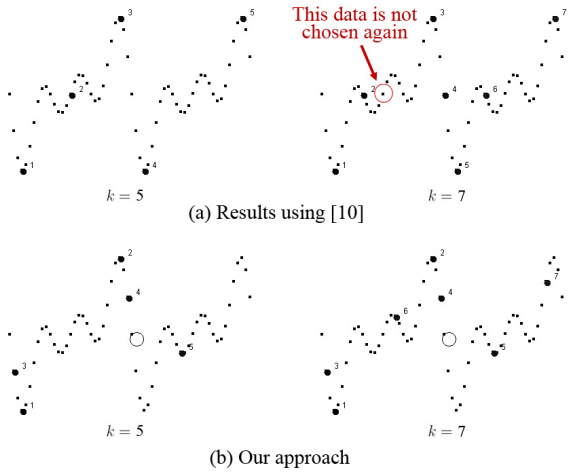


Fig. 3: A comparison of our approach and an optimization approach by [10] using $k = 5$ and $k = 7$

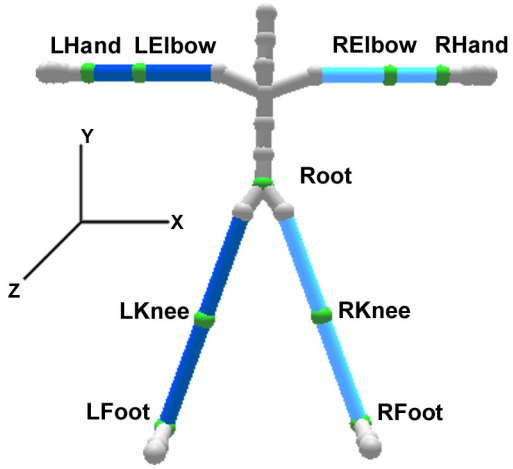


Fig. 4: Used joints

lating the keyposes. Unlike several previous approaches which also ignore the Y rotation of the root joint, we take all the rotations into account. Such approach allows motion segments that move along different directions to be recognized as different motion segments, for example if we have a walking motion which consists of a motion segment of walking to the east continued by walking to the west, then we can obtain the keyposes of both motion segments and show them to users so that they can understand that the motion represents a forward and backward walking motion, and not just a one directional walking motion. Further, as can also be seen in Figure 4, we select nine joints from all the joints of mocap data that can well represent human movement in daily activities, such as walking, running, and jumping, as elements in a vector V, which is described in section 3.

We compare our results to the results of [10] as described in the previous section. To highlight the differences, we intentionally choose a cyclic mocap data and a non-cyclic mocap data for comparison. As can be seen in the case of a cyclic mocap data in Figure 5, the approach in [10] allows the existences of multiple keyposes which are similar, while our approach focuses on obtaining keyposes, which are distinct. On the other hand, the results of both methods when applied to a non-cyclic mocap data do not show a lot of differences as can be seen in Figure 6.

Unfortunately, even in the case of cyclic motions, it is not self evident which results are superior and we believe that both methods are based on different objectives. While one particular application may prefer the existence of similar keyposes, another ap-

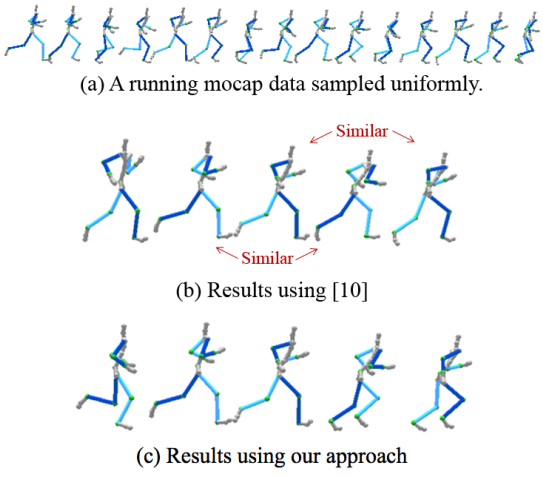
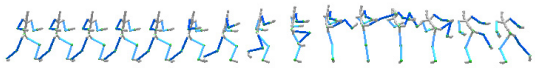
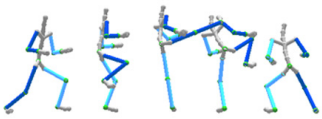


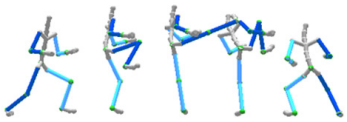
Fig. 5: Comparison for a cyclic mocap data.



(a) A kicking mocap data sampled uniformly.

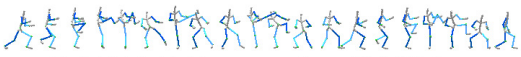


(b) Results using [10]

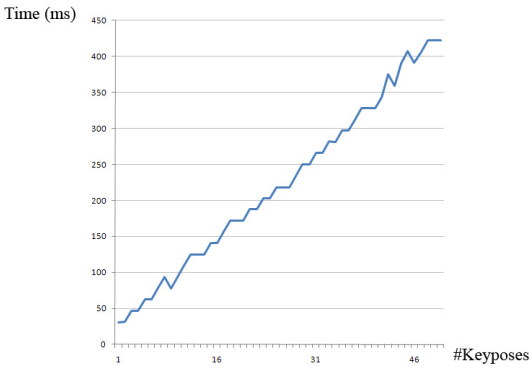


(c) Results using our approach

Fig. 6: Comparison for a non-cyclic mocap data.



(a) Multiple kicking mocap data with 1471 frames



(b) The time required to get keyposes

Fig. 7: Calculating the required time to extract keyposes.

plication may prefer to obtain only keyposes which have long distances to each other. For example, an application which focuses on displaying motion variations and the differences of a motion compared to the others may find our approach more attractive, while [10] may be favorable for applications which aim to show the motion periodicity.

As another experiment, we test our approach on a multiple kicking mocap data, which has 1471 frames, and calculate the required time to extract keyposes for numerous k values using an Intel Core Duo 1.86 GHz processor having 3GB memory. It can be seen that our approach is efficient as shown in Figure 7, and the required time can be approximated to grow linearly as the k value increases.

4.1 Manipulating the Number of Keyposes

For interactive applications, we believe that a user should be able to increase or decrease the number of keyposes as he/she wishes and be presented with the results very rapidly, since the appropriate number of keyposes depends on the content and the length of the mocap data itself. Recall that in our approach, all keyposes obtained by lower k values still become keyposes for higher k values. Thus, incrementing k only requires $O(n)$ operations to add one additional keypose, while decrementing k does not require any operation at all. On the other hand, optimization techniques such as [10] requires a full execution each time the number of keyposes is manipulated since keyposes for lower k values do not always become keyposes for higher k values.

4.2 Weighted Joints

Giving weights to the joints so that the distance of certain joints between poses is amplified or diminished is useful if a user wants to pay more attention to the movement of those joints. This can simply be obtained by giving weights to the joints. Since we require only $O(kn)$ operations, our approach allows a user to specify the joint weights and see the results very rapidly.

The results of giving more weights to a combination of hand and elbow joints, and a combination of knee and foot joints, can be seen in Figure 8. Figure 8 (a) shows us that giving more weights to hand and elbow joints will extract keyposes that prioritize the variations of hand and elbow joints instead of other joints. A similar case is also shown in Figure 8 (c) where more priority is given to the distances of the foot and knee joints when extracting keyposes.

4.3 The Effect of Global Translation in the Vertical Axis

Since we take global translation in the vertical axis into account in calculating keyposes, the change in root position will also cause the joint positions to change. This particular attribute will cause two poses which only have differences in their root positions to have longer distances because the differences of the joints are also added, even though the root is given the same weight as the other joints.

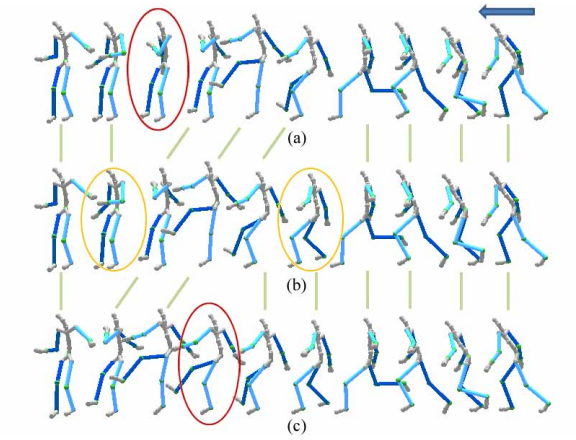


Fig. 8: A comparison of not using weights and using weights. Red and yellow circles highlight keyposes which are extracted and omitted respectively after adding weights. Two similar keyposes are connected with a green line. (a) More weights for hand and elbow joints. (b) No weights. (c) More weights for foot and knee joints

In other words, there is a *hidden* weight for the root.

To address this problem, we calculate the difference of joint positions, by first transforming them so that the corresponding roots match to each other, and then perform the distance calculation as illustrated in Figure 9. In Figure 9, the difference between pose A and pose B can be modeled as $(diff(root_a, root_b) + diff(joint_a, joint_{b'}))$, and not $(diff(root_a, root_b) + diff(joint_a, joint_b))$ which is commonly used, where $diff(x, y)$ measures the difference between x and y, for example by measuring their Euclidean distance.

By treating the problem this way, the hidden

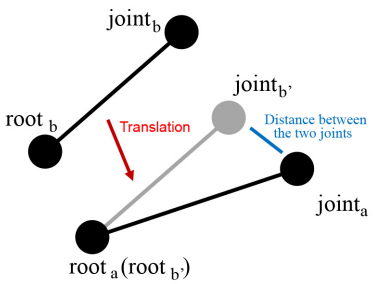


Fig. 9: Pose A $(root_a, joint_a)$ and pose B $(root_b, joint_b)$

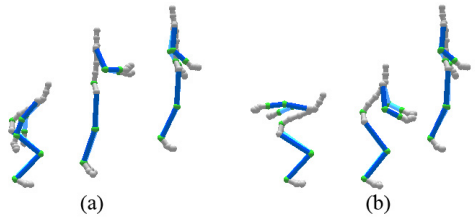


Fig. 10: Jumping motion (a) without matching and (b) with matching roots using $w_{root} = 5$.

weight of the root joint that previously existed is eliminated, and applying weights, as described in the previous subsection, can be used in order to give longer or shorter distances to the difference of root positions compared to the differences of the other joint positions.

In our experiment, giving $w_{root} = 5$, while letting the other weights equal to one generally yields satisfactory results as can be seen in Figure 10. For this configuration, frames which show the pose variations will be prioritized over frames which only convey the movement of the body, i.e the root joint, when extracting keyposes.

It should be noted that increasing the number of keyposes will generate similar results. We reduce the number of keyposes only to three in Figure 10 to demonstrate that more representative poses can be discovered earlier by applying this method.

Although we apply this method only to the root, it is pretty straightforward to apply the same method to other joints if we would like to use the local positions of the joints relative to their parents, instead of their local positions relative to the root for the calculation of keyposes. However, in selecting keyposes to represent movement in daily activities, we believe that this is not the general case and therefore refrain from doing so.

5 Discussion

In this paper, the authors have introduced a simple method to extract keyposes from mocap data. We have also shown that the computation from using the proposed method is very efficient, allowing users to increase or decrease keyposes as necessary, and at the same time allowing them to give weights to certain joints. We present the keyposes of several other complex motions in Figure 11. It can be seen in the figure that our method successfully obtains representative keyposes, and users will be able to understand what motion is contained in the mocap data by only looking at the keyposes. In addition,

the last motion in Figure 11, which is a punching and kicking motion, shows that our approach also works for mocap data which is composed of more than one motion segment.

As we pointed out in the previous section, our method is similar to [10] but the results are very different if the inputed mocap data is cyclic. We believe that both methods complement each other and one method may be more appropriate for certain applications than the other although further investigation is required to identify such applications. However, the complexity of our approach is the lowest possible given k keyposes and n data, and therefore we believe that our approach is applicable to most situations.

One weakness of our approach is that, since our approach works by first looking for the pose which is the furthest from the mean, it is weak against unsmoothed noise inside the data or unremoved defects from the recording of motion capture. To alleviate this problem, these noises must first be smoothed by using methods such as Gaussian filtering.

In this paper, the keyposes are represented in a side by side view, in which several images of one mocap data are arranged in a row, without considering the actual positions of the actor in the motion. Rendering the keyposes according to its actual positions yields plausible results if the actor is not staying at the same place, but confusing results if the actor maintains his/her position in a certain spot even if less important poses are rendered using higher transparencies. On the other hand, [2] introduced spatially expanded layout to place keyposes in a layout which preserves the overall structure of the activity instead of the original spatial location. However, this is a rather expensive operation since it requires an additional minimization operation. We believe that side by side view is able to show the relationship of one frame to the other frames well enough without consuming too much space on a user's display monitor. This can also be extended so that the distances between poses account for their distances in time as described in [7].

In the future, we are planning to combine our approach with a mocap data retrieval method, creating a system on the Internet that can search mocap data and display the search results effectively as in searching text documents, which should also allow users to upload their own mocap data to the system and process the uploaded mocap data efficiently, even during simultaneous upload. Such a system would allow other users who do not have access to mocap data devices to search mocap data rapidly and download the desired datasets, which can then be used to animate their own 3D models.

References

[1] Arikan, O., Forsyth, D. A., and O'Brien, J. F. Motion synthesis from annotations. *ACM Trans. Graph. 22*, 3 (2003), 402–408.

[2] Assa, J., Caspi, Y., and Cohen-Or, D. Action synopsis: pose selection and illustration. *ACM Trans. Graph. 24*, 3 (2005), 667–676.

[3] CMU. CMU graphics lab motion capture database. Retrieved November 2007, from http://mocap.cs.cmu.edu/, 2007.

[4] Forbes, K., and Fiume, E. An efficient search algorithm for motion data using weighted pca. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), ACM, pp. 67–76.

[5] Google. Google image search.

[6] Guralnik, V., and Srivastava, J. Event detection from time series data. In *KDD '99: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 1999), ACM Press, pp. 33–42.

[7] Hiroshi Yasuda, Ryota Kaihara, S. S., and Nakajima, M. Motion belts: visualization of human motion data on a timeline. *IEICE Transactions E91-D*, 4 (2007), 1159–1167.

[8] Huang, K.-S., Chang, C.-F., Hsu, Y.-Y., and Yang, S.-N. Key probe: a technique for animation keyframe extraction. *The Visual Computer 21*, 8–10 (2005), 532–541.

[9] Kovar, L., and Gleicher, M. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph. 23*, 3 (August 2004), 559–568.

[10] Morishima, S., Kuriyama, S., Kawamoto, S., Suzuki, T., Taira, M., Yotsukura, T., and Nakamura, S. Data-driven efficient production of cartoon character animation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches* (New York, NY, USA, 2007), ACM, p. 76.

[11] Müller, M., Röder, T., and Clausen, M. Efficient content-based retrieval of motion capture data. *ACM Trans. Graph. 24*, 3 (July 2005), 677–685.

[12] Perng, C.-S., Wang, H., Zhang, S. R., and Parker, D. S. Landmarks: A new model for similarity-based pattern querying in time series databases. *ICDE '00* (2000), 33.

[13] Tanuwijaya, S., and Ohno, Y. Data mining approach for positioning camera in displaying

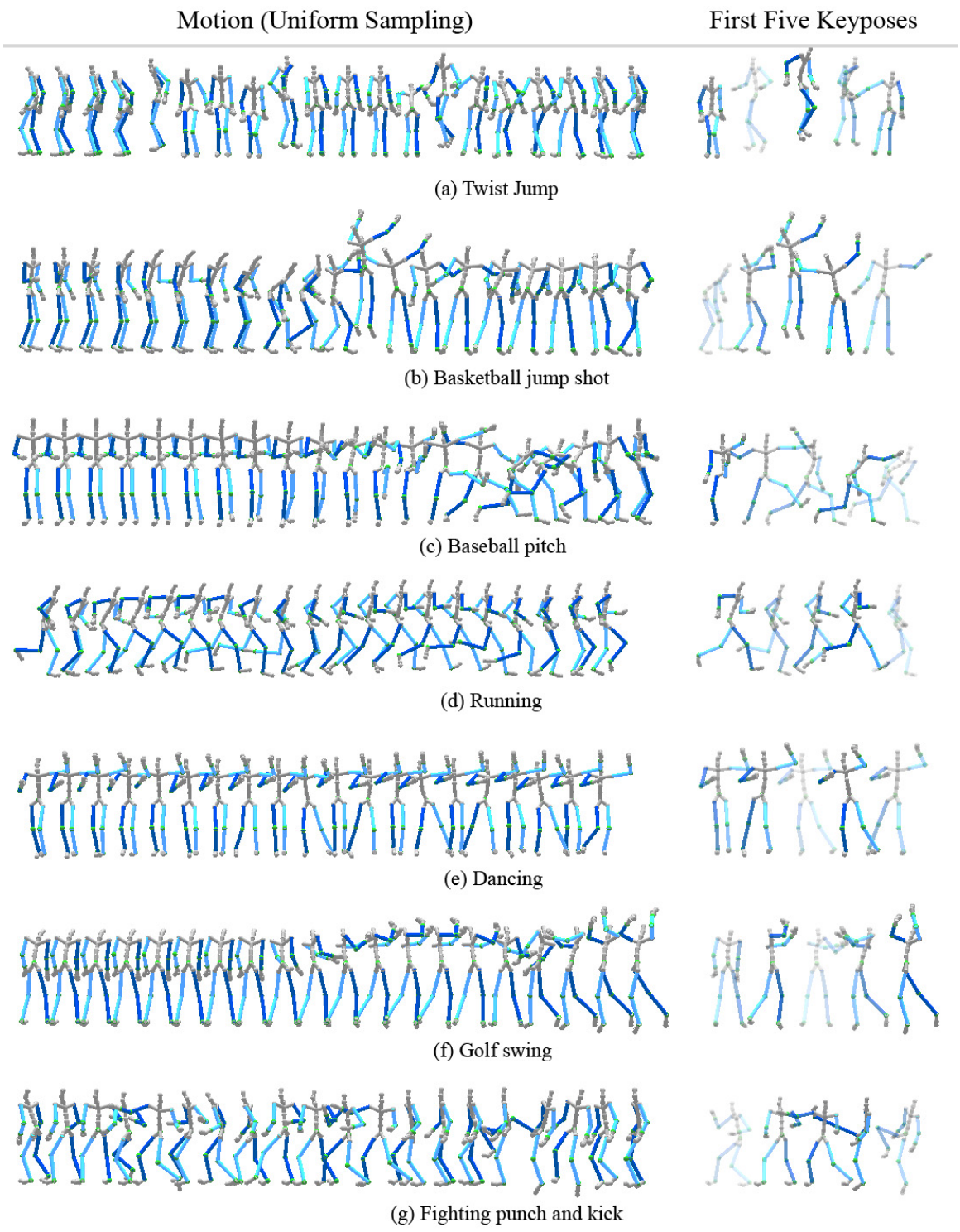


Fig. 11: Several mocap data and their first five selected keyposes. The order of the keyposes are denoted by the keyposes' transparencies.

mocap search results. In *Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications* (2009), INSTICC, pp. 266 – 273.

[14] Vlasic, D., Adelsberger, R., Vannucci, G., Barnwell, J., Gross, M. H., Matusik, W., and Popovic, J. Practical motion capture in everyday surroundings. *ACM Trans. Graph. 26*, 3 (2007), 35–43.