Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

ELSEVIER Computer Vision and Image Understanding 92 (2003) 265–284

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

3D motion retrieval with motion index tree

Feng Liu,* Yueting Zhuang, Fei Wu, and Yunhe Pan

Institute of Artificial Intelligence, Zhejiang University, Microsoft Visual Perception Laboratory of Zhejiang University, Hangzhou, 310027, PR China

Received 1 September 2002; accepted 1 June 2003

Abstract

With the development of Motion capture techniques, more and more 3D motion libraries become available. In this paper, we present a novel content-based 3D motion retrieval algorithm. We partition the motion library and construct a *motion index tree* based on a hierarchical motion description. The motion index tree serves as a classifier to determine the sub-library that contains the promising similar motions to the query sample. The Nearest Neighbor rule-based dynamic clustering algorithm is adopted to partition the library and construct the motion index tree. The similarity between the sample and the motion in the sub-library is calculated through elastic match. To improve the efficiency of the similarity calculation, an adaptive clustering-based key-frame extraction algorithm is adopted. The experiment demonstrates the effectiveness of this algorithm.

Keywords: Motion capture; Motion retrieval; Hierarchical motion description; Motion index tree; Dynamic clustering; Key-frame extraction; Elastic match

1. Introduction

Motion capture techniques have been widely used in computer animation, film making, game, etc. Optical, mechanical, or magnetic sensors are attached to the joints of a human performer to record his movement, which can be used to drive animated characters [1]. These techniques enable animators to produce realistic

* Corresponding author. Fax: +86-571-87951947.
E-mail addresses: lffred@yahoo.com.cn (F. Liu), yzhuang@cs.zju.edu.cn (Y. Zhuang), wufei@zju.edu.cn (F. Wu), panyh@sun.zju.edu.cn (Y. Pan).

1077-3142/$ - see front matter
doi:10.1016/j.cviu.2003.06.001

animations efficiently. With the popularity of motion capture systems, a large number of 3D motion libraries have been built.

However, problems occur when these 3D motion libraries are put into use, mainly on the following two aspects:

1. How to make full use of existing motions. Although motion capture systems can accurately record the motion of a performer, the captured motion does not necessarily meet the requirements of animators in practice. Animators can hardly predict exactly what motion they need, and even they can, they often change their intentions. Moreover, because animated characters often interact with the environment and other characters, existing motions need to be modified accordingly before applied to animated characters. In addition, the captured motions need to be adapted to the special configuration of different characters before retargeted to them.
2. How to get desired motions from the library. It is difficult and tedious for users to obtain required motions by browsing the library. The new animation technologies, such as [2], in which new motions are produced by extracting and synthesizing motions (fragments) similar to motion samples from the library, also demand the technique for retrieving motion data automatically.

In these years, fruitful work has been proposed to make full use of the captured 3D motion. A popular method is to adapt existing motions to new requirements through interactive control, such as constrained-based methods [3–5], motion retargeting [6], etc. Meanwhile, an attractive way is to apply techniques from signal and image processing fields to motion adaptation and new motion generation [7,8]. Recently, a more fantastic way is proposed to synthesize new motions from example [2,9–11]. That is to generate motions through selecting and piecing existing motions along a specified path or according to statistical distributions.

To the best of our knowledge, however, no available algorithms have been proposed to retrieve 3D motions based on their content. The most relevant work proposed [2] is to choose appropriate motion fragments from an existing motion library to synthesize required motions. This algorithm, however, is unsuitable for the full-body motion retrieval, as motion match is executed by comparing partial scripted key-frame motions with partial captured motions.

Inspired by content-based retrieval algorithms for retrieving multimedia data based on the features automatically extracted from the content [12–15], we propose a content-based motion retrieval algorithm. Particularly, desired motions are obtained by submitting a similar sample in the form of a captured motion or a scripted one. To achieve this goal, the following problems have to be addressed:

1. An effective motion representation is needed. Each motion is a frame sequence, with each frame defining a posture. Because each posture is a configuration made up of all the body joints and each motion is a harmonic combination of sub-motions of all these joints, an efficient description of the posture is required. The description should also address the different effects of each joint on determining the posture.
2. An efficient match algorithm is demanded to calculate the similarity between two motions. Each posture in a motion is defined by a frame with tens of parameters

depicting the character configuration, and the temporal order among the posture/frame sequence is essential, so an efficient and effective match algorithm is demanded.

3. A key-frame extraction algorithm is required. Because each motion consists of a large number of frames, it is time-consuming to calculate the similarity or distance between two motions with these original data. To reduce the computational overhead, key-posture sequences need to be extracted to calculate the similarity.

Aiming at the above challenges, we present a content-based 3D motion retrieval algorithm. Specifically, we describe two main stages in this paper, first building a motion index tree to structure the motion library, and then retrieving motions using the motion index tree. Particularly, we adopt a hierarchical motion description to represent a posture, based on which we partition the motion library hierarchically and construct a motion index tree to facilitate motion retrieval. Nearest Neighbor rule-based dynamic clustering algorithm is employed to partition the motion library and construct the motion index tree. The similarity between two motions is calculated by elastic match. To improve the efficiency during the similarity calculation, we adopt an adaptive clustering-based key-frame extraction algorithm to extract key-posture/-frame sequences, which are used in elastic match. During the retrieving stage, the motion index tree serves as a hierarchical classifier to determine the sub-library that contains the promising similar motions to the query sample. Next, key-frame/-posture sequences are extracted from the sample and the motion from the sub-library respectively, and the similarity between them is calculated using elastic match.

The remainder of this paper is organized as follows. In the next section, we give a review on previous work. In Section 3, we give a brief description on the hierarchical motion representation. In Section 4, we describe the construction of the motion index tree in detail. In Section 5, we describe the procedure of motion retrieval. We discuss the experiment in Section 6 and conclude the paper in the last section.

2. Previous work

In this section, we first make a brief review on previous work on motion processing, namely motion editing, motion analysis, and synthesis. Then we summarize related work with key-posture/key-frame extraction used in the paper.

2.1. Motion editing, analysis, and synthesis

Early research on motion processing aimed to provide convenient tools for interactive motion editing. A popular way is constraint-based method. Gleicher [3] and Gleicher and Litwinowicz [4] proposed a space-time constraint-based method for editing a pre-existing motion such that it meets various requirements yet preserves its original quality as much as possible. However, it is time-consuming to solve the constraint optimization problem in this method. To improve the efficiency, Lee and Shin [5] presented a hierarchical approach, which divides the constraint optimization

problem into two sub-processes: (1) the configuration of an articulated figure is adjusted to meet the constraint on each frame by an inverse kinematics solver and (2) the motion displacement in each constrained frame is interpolated and thus smoothly propagated to the nearby frames using a fitting technique. Another attractive method is the so-called motion signal processing. Bruderlin and Williams [7] applied techniques from image and signal-processing fields to designing, modifying, and adapting motions. Similarly, Liu et al. [8] provided a series of tools for editing motion at a high level by introducing wavelet transformation into motion analysis and synthesis.

Recently, a fantastic technique, example-based motion synthesis, is proposed. Lee et al. [9] developed a technique for controlling an animated character in real time using several possible interfaces, through which users can choose from a set of possible actions, sketch a path on the screen and create animations by searching through a motion database using a clustering algorithm. The approach of Li et al. [10] combines some low-level noise driven motion generators with a high-level Markov process to generate new motions with variations in fine details. Pullen and Bregler [2] presented a motion capture assisted animation, which allows animators to key-frame motions for a sub-set of Degree of Freedoms (DOFs) of a character and use motion capture data to synthesize motion for the missing DOFs and add texture to those key-framed. Kovar et al. [11] build a motion graph that encapsulates connections among the motion library and synthesize motions by building walks on the graph.

2.2. Key-frame extraction

Because motion resembles video in their representation (i.e., both of them can be represented as posture/frame sequences), we explore key-frame extraction algorithms in video abstraction to find or adapt an appropriate one for extracting key-frames from motions. Below we give a review on key-frame extraction from video.

In the earlier work such as [16,17], key-frames are selected by sampling video frames randomly or uniformly at certain time intervals. Though this is a fast way to extract key-frames, it neglects the actual video content. Therefore, many representative frames are missing, especially from short segments, whereas redundant frames are extracted especially from long segments. To address this problem, shot-based key-frame extraction algorithms are proposed. A video clip is first segmented into shot sequences based on features such as color [15], motion [18,19], and then a certain frame in each shot, e.g., the first or the last frame, is selected as a key-frame. Another way is sequential comparison-based method [20], in which the current frame is compared with the last extracted key-frame. If the difference is significant, then the current frame is selected as a new key-frame. To avoid successive selection in highly active frames, a minimal interval between key-frames is set [21]. A special way is to utilize the visual content as well as such information as corresponding audio streams for key-frame detection [22,23].

As addressed in Kim and Hwang [24], previous key-frame extraction algorithms relied mostly on low-level features and other readily available information instead of using semantic primitives. Recently, object-based video abstract techniques have

been presented in [25–27]. The representative work of object-based algorithms is reported in Kim and Hwang [21,27]. The objects in each frame are first segmented and identified through a moving-edge detection algorithm [28,29], and the first frame in each shot is selected as the first key-frame. If the object number changes, the current frame is selected as a new key-frame. Otherwise the dissimilarity between the current frame and the recent extracted key-frame is calculated through measuring the dissimilarity between two object masks extracted from the two frames, respectively, and if it excesses a given threshold, a new key-frame is created. These object-based methods work especially well in video surveillance systems. However, they are not suitable for extracting key-frames from motion, as there is no object, even no background/foreground in motion at all.

Because a large number of motions are nearly periodical, a promising method is clustering-based extraction. Particularly, similar frames are clustered into the same cluster, and a representative frame from each cluster is selected as a key-frame. Ratakonda et al. [30] proposed a hierarchical video summarization using a pair-wise K-means algorithm. However, this temporal constrained K-means clustering cannot merge similar but temporally apart frames. Doulamis et al. [31] adopted a fuzzy classifier to cluster all features extracted through a recursive shortest spanning tree algorithm to predetermined classes. And a genetic algorithm is adopted to extract key-frames by minimizing a cross-correlation criterion. This method is highly time-consuming. Kim and Hwang [24] also present an object-based video abstraction through Mean Shift Clustering.

3. Motion representation

Motion can be described as a frame sequence, with each frame depicting a posture at a given time. An intuitive description of a posture is a set of bones, each represented as a rigid 3D model with a position and orientation, together with a set of constraints imposed on joints that prevent them from separating or rotating in illegal ways. Within this model, the DOFs of a character are the positions and orientations of bones in the body.

A more concise representation imposes a hierarchical relationship on the bones, in which the position and orientation of each bone are specified with regard to its ''parent'' bone. A bone's configuration at a specific time can be specified by a fixed translation (thus not involving any DOFs) and a rotation with regard to its parent bone, which is in turn defined in terms of its parent. The configuration is recursively defined all the way up to the root joint, which has full translational and rotational DOFs [32]. In the case of a human body, the posture can be depicted as a tree, as illustrated in Fig. 1. Within this hierarchical posture description, the DOFs of a character include the rotational parameters of non-root joints and the position and orientation of the root joint (which is *Pelvis* in Fig. 1). Each motion can be represented as follows:

$$M(t) = (T_{\text{root}}(t), Q_{\text{root}}(t), R_1(t), R_2(t), \ldots, R_n(t)), \tag{1}$$

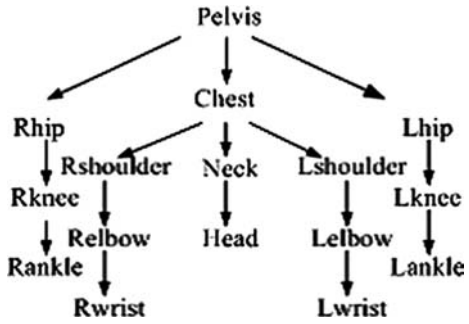

Fig. 1. Hierarchical motion description.

where $T_{\text{root}}(t)$ and $Q_{\text{root}}(t)$ are the position vector and orientation vector of root at time t, and $R_i(t)$ is the rotation vector of joint i around its parent joint at time t.

This hierarchical description implicitly holds the bones together. Moreover, it clearly demonstrates the different role played by each joint in determining the posture, with parent joints being more prominent than children joints. Furthermore, it reflects the effect of each joint's motion on the full-body motion, with the motions of parent joints being more significant than those of children joints. Motions represented in the intuitive way can be transformed into the hierarchical description using inverse kinematics [32].

4. Motion index tree

Within the hierarchical motion description, the motion of a parent joint may induce those of its children joints, whereas that of a child joint is unable to influence its parent joint. Obviously, the joints at high levels of the hierarchy are more significant than those at lower levels in terms of determining a motion. This hierarchy among the parameters of a posture can be well used to facilitate motion retrieval by building a corresponding motion index tree for a motion library.

Given the above human body as an example, all the joints can be divided into the following five levels according to their positions in the tree illustrated in Fig. 1, as {Root}, {Lhip, Rhip, Chest}, {Lknee, Rknee, Neck, Lshoulder, Rshoulder}, {Lankle, Rankle, Head, Lelbow, Relbow}, and {Lwrist, Rwrist} from top to bottom. We construct a motion index tree based on this hierarchy to partition and structure the 3D motion library, as shown in Fig. 2. This motion index tree serves as a hierarchical classifier to determine the sub-set that contains promising similar motions to a submitted example. For this sake, each non-leaf node in the motion index tree contains a sample set, built by selecting representative motions from its children nodes as described below. Each sample is labeled with the information of its source. The sample set is used in kNN rule (k Nearest Neighbor rule) to classify the submitted example in the process of retrieval described in Section 5. Each leaf node is associated with a sub-set, which contains promising similar motions to the example. All these sub-sets

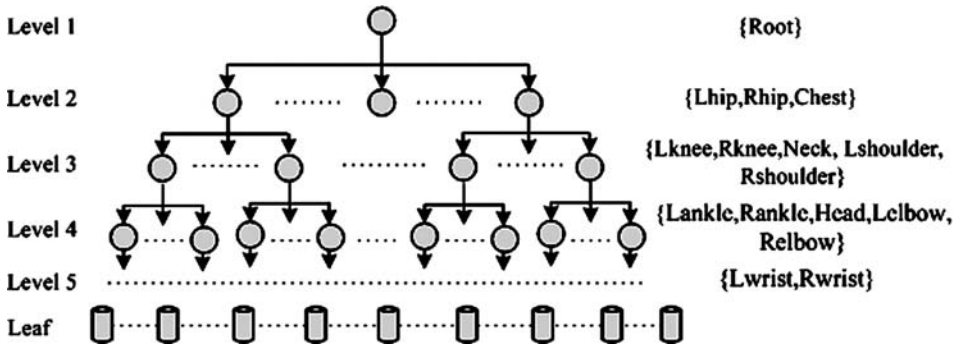

Fig. 2. Motion index tree.

constitute the whole motion library. (For simplicity, the sub-motions of the joints at one or multiple levels are also called motions in the subsequent sections.)

The main steps to construct the motion index tree are outlined as follows:

1. Partition the 3D motion library $\boldsymbol{ML}$ into several sub-sets $\boldsymbol{ML}_{2,k}$ using dynamic clustering (to be described in Section 4.1) according to the motion of the first-level joint (viz. {Root}), and build a sample motion set from the sub-sets for the node Root. Particularly, motions closest to the centroid of each sub-set are selected as samples from this sub-set. Create a null node with an empty sample set, associate it with a sub-set $\boldsymbol{ML}_{2,k}$, and take it as one of the children nodes of Root.
2. Partition each sub-set $\boldsymbol{ML}_{i,k}$, respectively, according to the motion of joints at the level i in the hierarchy and build a sample set for its corresponding node in the motion index tree in the similar way in Step (1). Again, create a null node for a newly created sub-set $\boldsymbol{ML}_{i+1,j}$ and take it as a child node of $\boldsymbol{ML}_{i,k}$.
3. Repeat Step (2) until all joints at the lowest level of the hierarchy are processed.
4. Take the last partitioned sub-sets as leaf nodes of the motion index tree.

In the following subsections, we will first describe the dynamic clustering algorithm for partitioning the motion library, and then elaborate the key-frame extraction algorithm and the elastic motion match algorithm, respectively.

4.1. Nearest Neighbor rule-based dynamic clustering

Partitioning a motion library into sub-sets is a well-defined clustering problem. As described in Section 3, motion is a high-dimensional signal and no function is available to describe the probability density up to now. Due to the lack of available probability density functions, we partition the motion library based on the similarity between sample motions. Therefore, Nearest Neighbor rule-based dynamic clustering [33] is adopted to partition and structure the library.

Given two motion samples M_i and M_j, if M_j is the Ith Nearest Neighbor (NN) of M_i, the "NN coefficient" of M_j to M_i is defined as I. Likewise, if M_i is the Kth Nearest Neighbor of M_j, the NN coefficient of M_i to M_j is defined as K. Let α_{ij} be the

"NN value" between M_i and M_j, $\alpha_{ij} = I + K - 2$. If M_i and M_j are classified into the same cluster, the connection cost is defined as α_{ij}; otherwise, the cost is 0. To eliminate clusters with only one sample, the connection cost of the self-connection is defined as $2N_M$ (N_M is the number of motions in the library).

The objective function J_{NN} is defined as the sum of the total inner-cost L_{IA} and total inter-cost L_{IR}:

$$J_{NN} = L_{IA} + L_{IR}.$$

The total inner-cost L_{IA} can be defined as the sum of all the connection values between every pair of samples in the whole library, given that the connection cost between samples from different clusters is 0.

To calculate the inter-cost, the minimal NN value between cluster ω_i and ω_j, γ_{ij}, is computed first. And γ_i, the minimal NN value between cluster ω_i and all the other clusters can be calculated as follows:

$$\gamma_i = \min_{j \neq i} \gamma_{ij}.$$

Let $\alpha_{i\max}$ and $\alpha_{k\max}$ be the maximal connection cost between samples in the cluster ω_i and ω_k each, c be the number of clusters, β_i, the inter-cost of cluster ω_i to the other clusters, is defined as follows:

$$\beta_i = \begin{cases} -[(\gamma_i - \alpha_{i\max}) + (\gamma_i - \alpha_{k\max})] & (\gamma_i > \alpha_{i\max},\ \gamma_i > \alpha_{k\max}), \\ \gamma_i + \alpha_{i\max} & (\gamma_i \leqslant \alpha_{i\max},\ \gamma_i > \alpha_{k\max}), \\ \gamma_i + \alpha_{k\max} & (\gamma_i > \alpha_{i\max},\ \gamma_i \leqslant \alpha_{k\max}), \\ \gamma_i + \alpha_{k\max} + \alpha_{i\max} & (\gamma_i \leqslant \alpha_{i\max},\ \gamma_i \leqslant \alpha_{k\max}), \end{cases}$$

$$k = \arg \min_{j \neq i, j \in \theta, \theta = \{1,2,\ldots,c\}} \gamma_{ij}.$$

The total inter-cost can be defined as the sum of all the inter-costs.

The optimal clustering should result in the minimal objective value J_{NN}. The following iterative method is proposed to solve this problem:

1. Calculate the distance matrix Δ, with each element $\Delta_{ij} = \Delta(\boldsymbol{M}_i, \boldsymbol{M}_j)$, where $\Delta(\boldsymbol{M}_i, \boldsymbol{M}_j)$ is the distance between motion $\boldsymbol{M}_i$ and $\boldsymbol{M}_j$. The distance calculation is described in Section 4.3.
2. Construct NN coefficient matrix $\boldsymbol{M}$ based on Δ, with each element $\boldsymbol{M}_{ij}$ being the NN coefficient of $\boldsymbol{M}_i$ to $\boldsymbol{M}_j$.
3. Build NN value matrix $\boldsymbol{L}$ based on the NN coefficient matrix $\boldsymbol{M}$

$$L_{ij} = \begin{cases} M_{ij} + M_{ji} - 2 & (j \neq i), \\ 2N_M & (j = i). \end{cases}$$

4. Connect each element to the element, to which it has the minimal NN value in $\boldsymbol{L}$, and form the initial clusters.
5. Calculate γ_i of each cluster and compare it with $\alpha_{i\max}$ and $\alpha_{k\max}$. If γ_i is smaller than either of $\alpha_{i\max}$ and $\alpha_{k\max}$, merge cluster ω_i and ω_k.
6. Repeat Step 5 until no clusters can be merged.

4.2. Key-frame extraction

Because each motion is represented as a frame sequence, the similarity between two motions is defined as that between the two corresponding frame sequences. However, even a short motion of 4 s is composed of approximately 100 frames. It is time-consuming to calculate the similarity with these original data. To improve the computational efficiency, we extract key-frame sequences from two motions and calculate the similarity between them as that between the motions. Taking the motion of the second level joints as an example, we show the key-frame extraction algorithm in detail below.

Given a motion M with N frames, the motion at the second level can be represented as a $(3 \times 3 \times N) \times 1$ vector M_2:

$$\begin{aligned} M_2 &= [F_1, F_2, \ldots, F_N], \\ F_i &= [r_{lxi}, r_{lyi}, r_{lzi}, r_{rxi}, r_{ryi}, r_{rzi}, r_{cxi}, r_{cyi}, r_{czi}], \end{aligned} \tag{2}$$

where F_i is the ith frame at the second level, r_{lxi}, r_{lyi}, and r_{lzi} are the rotational parameters of joint Lhip, r_{rxi}, r_{ryi}, and r_{rzi} are the rotational parameters of joint Rhip, and r_{cxi}, r_{cyi}, and r_{czi} are the rotational parameters of joint Chest.

As presented in Section 2.2, many methods have been proposed for key-frame extraction in the field of video abstraction. However, due to the periodicity of motions, most of these algorithms, including sampling-based, shot-based, and sequential comparison-based methods, will cause the redundancy in that similar frames in different cycles are extracted as key-frames. Methods, in which other information, such as audio stream, objects in each frame, are utilized for key-frame detection, are obviously not suitable for key-frame extraction from motions, as the only information available is the posture sequence in each motion.

A promising way is clustering-based extraction [24,30,31]. Particularly, similar frames are clustered into the same cluster, and a representative frame from each cluster is selected as a key-frame. We adopt the adaptive clustering-based key-frame extraction technique proposed in [34], in which similar frames in different cycles can be clustered into the same cluster.

Define the similarity between two frames as some decreasing function f_d about the weighted distance between them:

$$Sim(F_1, F_2) = f_\mathrm{d}\left(\sqrt{\sum_k w_k (F_{1k} - F_{2k})^2}\right), \tag{3}$$

where F_{1k} and F_{2k} are the motion parameters of frame F_1 and F_2, respectively and w_k is the weight indicating the significance of joint k. If these joints are from different levels in the hierarchy illustrated in Fig. 2, we give higher weights to joints at higher levels whereas lower weights to those at lower levels empirically. Let δ_i be the ith cluster, the clustering algorithm can be summarized as follows:

1. Initialization: $F_1 \to \delta_1$, $F_1 \to F_{c1}$, the centroid of δ_1, $1 \to$ numCluster.
2. Get the next frame F_i. If the frame pool is empty, end.

3. Calculate the similarity between F_i and the centroid of an existing cluster δ_k ($k = 1, 2, \ldots$ numCluster) according to Eq. (3).
4. Determine the cluster closest to F_i through calculating *Maxsim* as follows:

$$Maxsim = \max_{k=1}^{\text{numCluster}} sim(F_i, F_{ck}). \tag{4}$$

If *Maxsim* is below a given threshold, it means F_i is not close enough to be put into any cluster, goto Step 5; otherwise put F_i into the cluster with *Maxsim*, and goto Step 6.

5. numCluster = numCluster + 1. A new cluster is created: $F_i \rightarrow \delta_{\text{numCluster}}$.
6. Update the cluster centroid as follows:

$$F_{ck} = \frac{nF'_{ck} + F_i}{n+1}, \tag{5}$$

where F'_{ck} and F_{ck} are the centroids before and after update, respectively, and n is the size of the old cluster. Goto Step 2.

According to Zhuang et al. [34], the frame that is closest to the centroid of a cluster is selected as a key-frame. However, as the order among frames is lost during clustering, the extracted key-frame sequence is not consistent with the original temporal sequence. To preserve this essential attribute, considering frames in the same cluster are extracted sequentially, the first frame in each cluster is selected as the key-frame. The effectiveness of this algorithm will be discussed in Section 6.1. This algorithm can also be applied to the motion with multiple level joints or to the full body motion.

4.3. Motion match

According to the hierarchical motion description in Section 3, the global position of a body is determined by $T_{\text{root}}(t)$ in Eq. (1). Because the initial position of a motion is inessential in comparing one motion with another, we replace the absolute position of a motion with its velocity to eliminate the disturbance of the initial position yet preserve the information of the global locomotion.

Most similarity measures for motions are defined based on their corresponding key-frame sequences. A simple method is the Nearest Center (NC) algorithm. Zhang et al. [15] proposed a Nearest Neighbor approach, in which the similarity is defined as the sum of the most similar pairs of key-frames. As discussed in Li et al. [35], these methods neglected the temporal order of frames. To address the problem, Li [35] presented a Nearest Feature Line-based method. Though this method accommodated the temporal correlation between key-frames, it could only be applied to retrieving motions using a single frame as the query sample.

Elastic match performed with a dynamic time warping algorithm is a non-linear match method originally used in speech recognition and it has been successfully applied to online signature verification [36]. Elastic match can be used in comparison of all kinds of continuous function about a continuous parameter, which is time typically.

Given two motions, $M1$ $\{F1_1, F1_2, \ldots, F1_N\}$ and $M2$ $\{F2_1, F2_2, \ldots, F2_M\}$, the distance between them is defined as follows:

$$D = \frac{1}{2}\left(\min_{\{\omega 1(i)\}} \sum_{i=1}^{N} d(i, \omega 1(i)) + \min_{\{\omega 2(i)\}} \sum_{i=1}^{M} d(i, \omega 2(i))\right), \tag{6}$$

where the first factor on the right part is the distance between $M1$ and the motion resulting from warping $M2$ according to the path defined by a time warping path $\omega 1(i)$, and the second factor on the right is the distance between $M2$ and the motion resulting from warping $M1$ according to the path defined by $\omega 2(i)$. $d(i,j)$ is the distance between the ith frame of $M1$ and the jth frame of $M2$, defined as:

$$d(i,j) = \sqrt{\sum_k w_k (F1_{ik} - F2_{jk})^2}, \tag{7}$$

where $F1_{ik}$ and $F2_{jk}$ are the kth motion parameters in frame $F1_i$ and $F2_j$, respectively, andw_k is the weight, set in the same way described in Section 4.2.

The time warping path, for example, $\omega 1(i)$, is constrained by the following boundary and continuity conditions. The boundary conditions ensure that the first and last frame of $M1$ are matched to the frame b and frame e of $M2$: $\omega 1(1) = b$, $\omega 1(N) = e$, where $b = \min_{i \leqslant M} \arg(d(1,i) \leqslant \text{threshold})$ and $e = \max_{i \leqslant M} \arg(d(N,i) \leqslant \text{threshold})$.

The continuity conditions restrict the match of the intermediate frames, and $\omega 1(i)$ is defined as a monotonically increasing function and thus the temporal order is preserved during match.

Let the left half part of D be D_L, defined as $D_L = \min_{\{\omega 1(i)\}} \sum_{i=1}^{N} d(i, \omega 1(i))$, we solve it recursively by applying dynamic programming in the following way [37]:

$$\begin{aligned} &D_L(i,j) = d(i,j) + \min\{D_L(i-1,j), D_L(i-1,j-1), D_L(i-1,j-2)\}, \\ &D_L(1,b) = d(1,b), \end{aligned} \tag{8}$$

where $D_L(i-1, j-2)$ corresponds to skipping the $(j-1)$th frame of $M2$ and $D_L(i-1,j)$ means that at least two frames of $M1$ correspond to the jth frame of $M2$.

5. Motion retrieval

By now, the motion index tree has been constructed. We describe motion retrieval using the motion index tree in this section.

We devise a two-stage process to retrieve similar motions to a query example M. Determine a sub-set that contains promising similar motions to M and calculate distances/similarities between M and motions in the sub-set. The process of motion retrieval is outlined as follows:

1. Fetch M_1, the motion of the first-level joints, from the example M.
2. Extract key-frame sequences from $M1$ and each motion in the sample set of Root in the motion index tree using the key-frame extraction described in Section 4.2, and calculate the distances between them using elastic match described in Section 4.3. Then get the k Nearest Neighbors. Let $k_1, k_2, \ldots, k_c$ be the number of the

Nearest Neighbors belonging to the children nodes of Root $\omega_1, \omega_2, \ldots, \omega_c$, respectively, the decision-making function is defined as $g_i(M_1) = k_i,\ i = 1, 2, \ldots, c$ and the decision-making rule is defined as follows:

If $g_j(M_1) = \max_i k_i,\ i = 1, 2, \ldots, c$, then M_1 belongs to ω_j, the jth child node of Root.

3. Continue classification until a leaf node of the motion index tree is reached. Calculate the similarity between M and each motion stored in the sub-set in the leaf node, and select appropriate motions as the result according to the similarity.

6. Experiment and discussion

To validate the effectiveness and efficiency of the proposed technique, we develop a prototype system of content-based motion retrieval (CBMR) and test it on a motion library consisting of about 450 different motions. The composition of the library is shown in Table 1.

6.1. Key-frame extraction

Both the adaptive clustering-based key-frame extraction algorithm and the sequential comparison-based method, in which a frame is selected as a new key-frame when the difference between this frame and the last key-frame is significant, are implemented and tested. For convenience, these two algorithms are called ACE and CE, respectively. In both algorithms, the similarity between two frames is defined as the reciprocal of the weighted distance between them.

First, two non-periodical motions, *jump-kick* and *dive*, are experimented on. *jump-kick* consists of 70 frames and *dive* consists of 50 frames. The *Maxsim* value in ACE and the similarity in CE are shown in Fig. 3. For ACE, when the *Maxsim* value is below a given threshold, a new cluster is found and the current frame is selected as a new key-frame. From Fig. 3, we can see that totally 11 key-frames, including the first frame and the other 10 new key-frames, are extracted from *jump-kick*, and totally 10 key-frames, including the first frame and the other 9 new key-frames,

Table 1
Composition of the motion library

Motion class	Size of class	Motion class	Size of class
Walk forward	70	Rotate	32
Run forward	62	Run backward	32
Walk backward	54	Climb	28
Jump	48	Box	18
Squat	45	Wave hand	12
Ballet dancing	36	Fall	12

Eight students are engaged in partitioning the library. A motion can be classified into a certain class only when it gains ratifications from more than five of these students.

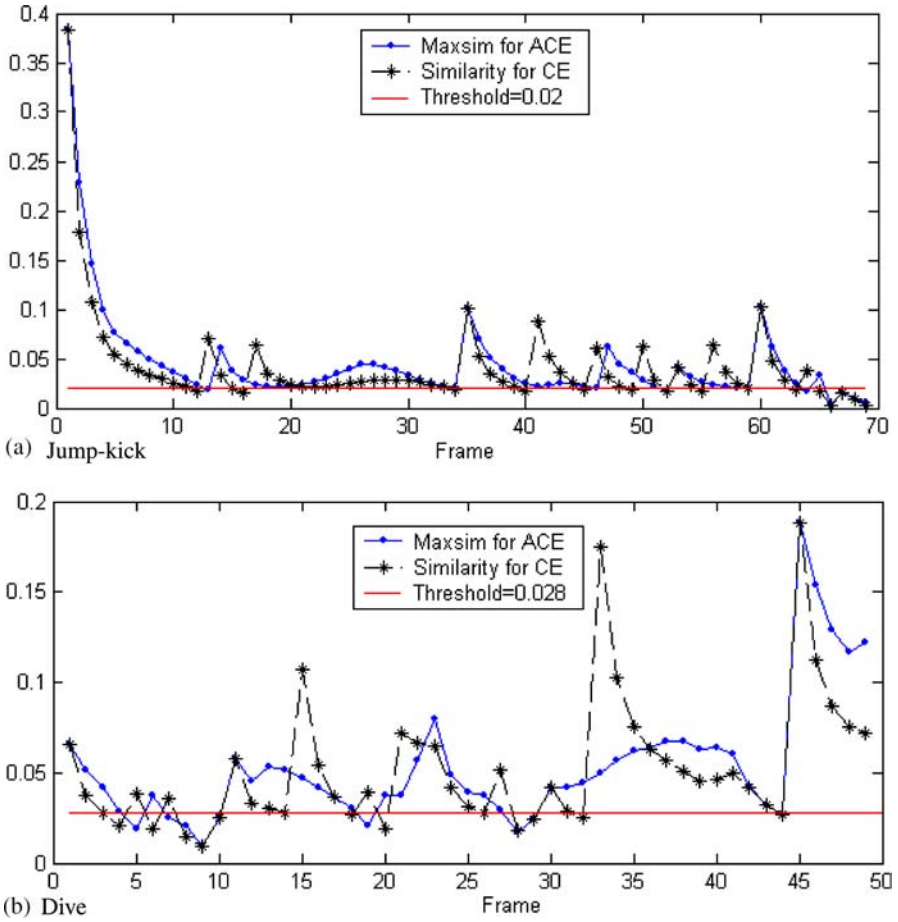

Fig. 3. Extract key-frame sequences from non-periodical motions using ACE and CE.

are extracted from *dive*. For CE, the frame with its similarity below a given threshold is selected as a new key-frame. From Fig. 3, we can see that totally 16 key-frames, including the first frame and the other 15 new key-frames, are extracted from *jump-kick*, and totally 14 key-frames, including the first frame and the other 13 new key-frames, are extracted from *dive*. The extracted key-frame sequences are shown in Fig. 4. As shown in Fig. 3, more key-frames are extracted using ACE than using CE from both motions given the same threshold though the same similarity measure is adopted. Two reasons account for it. The *Maxsim* value in ACE is defined as the maximal similarity between the current frame and the centroid of each cluster, which means that the current frame is compared with the centroids of all the previously extracted clusters. In CE, the current frame is only compared with the most recent key-frame, which rules out the possibility for the current frame to match

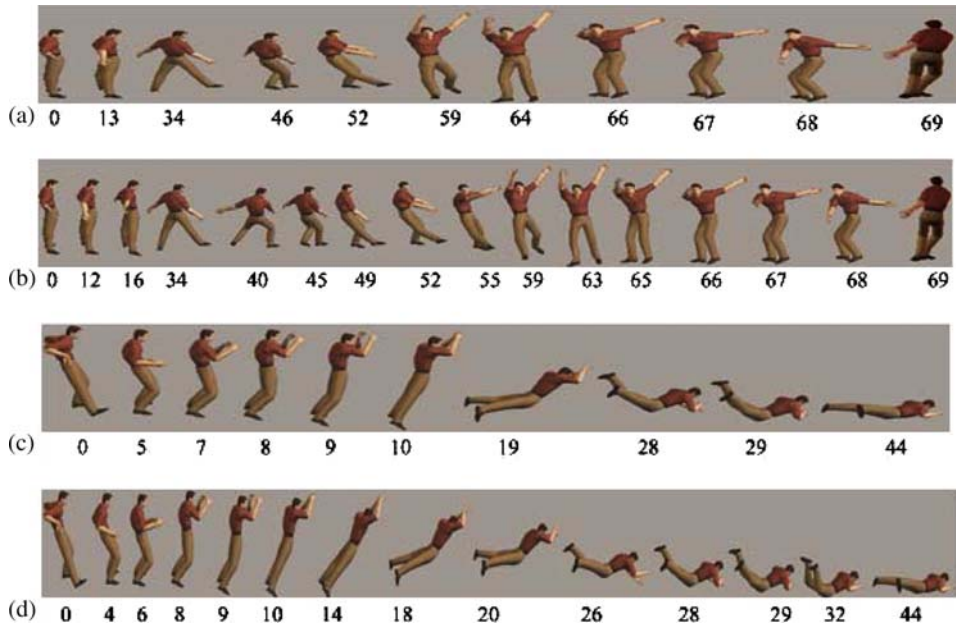

Fig. 4. Key-frame sequences extracted from non-periodical motions. (a) Key-frame sequence extracted from *jump-kick* by ACE. (b) Key-frame sequence extracted from *jump-kick* by CE. (c) Key-frame sequence extracted from *dive* by ACE. (d) Key-frame sequence extracted from *dive* by CE.

the most similar key-frame previously extracted. Another reason is that each motion is continuous, so the frames nearby are usually classified into the same cluster. Then the similarity between the current frame and the centroid decreases more slowly in ACE than that between the current frame and the last extracted key-frame in CE, which results in the fact that in ACE more frames after the last extracted key-frame have similarities over the threshold than in CE. As shown in Fig. 5, the similar key-frame sequences can be extracted by ACE at a relatively high threshold and by CE at a relatively low threshold. These results show that the performance of CE and ACE are similar for non-periodical motions.

Second, two periodical motions, *power-walk* and *hurdle*, are tested on. *power-walk* consists of 120 frames, and *hurdle* consists of 195 frames. *power-walk* is composed of about 4 cycles of steps. From Fig. 6a, we can see that using ACE, 5 clusters are formed in *power-walk*, including the initial one and the appended 4. The first frame of each cluster is extracted as a key-frame. All these key-frames are extracted from the first 30 frames, which compose the first cycle of *power-walk*. No frames in the successive cycles are extracted as key-frames. This shows the compactness of the extracted key-frame sequence. When it comes to CE, totally 29 key-frames, including the initial frame and the appended 28, are extracted from *power-walk*. From Fig. 6a, we can see that the last 28 key-frames can nearly be divided into 4 cycles and the last 3 seven-frame sequences are almost the repetitions of the first seven-frame sequence.

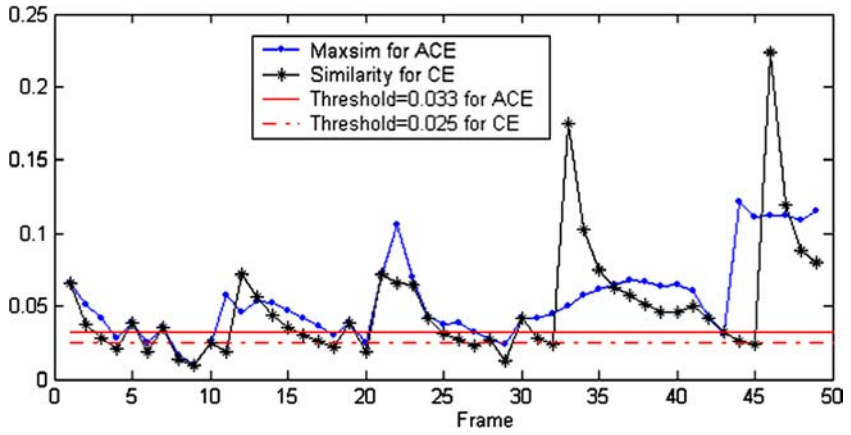

Fig. 5. Key-frame extraction using ACE and CE with different thresholds. The key-frame sequence extracted from dive by ACE with threshold, 0.033, is 0 4 6 8 9 10 18 20 27 28 29 43, and the key-frame sequence extracted by CE with threshold, 0.025, is 0 4 6 8 9 11 18 20 27 29 32 45.

The experiment on *hurdle* shows the similar result. All these demonstrate that ACE is superior to CE when applying to periodical motions.

6.2. Motion match

To evaluate the performance of the elastic match algorithm, we implement this algorithm, as well as other methods, such as the NC method and the NN method, in our CBMR system. Some motions are selected from the library shown in Table 1 as examples for queries to evaluate each algorithm. The distribution of these examples is stated in Table 2. Based on these queries, we plot the average precision-recall graph for each algorithm as illustrated in Fig. 7, from which we can see that elastic match is superior to both NC and NN. The main reason is that both NC and NN neglect the temporal order among the frame sequence of a motion. For example, *walk* cannot be distinguished from *run*, and *walk-back* cannot be distinguished from *walk-forward* either. Contrary to NC and NN, the continuity of the motion is preserved during the process of elastic match as a result of the constraint of continuity as addressed in Section 4.3. The result of one query is shown in Fig. 8. Each row represents a motion. The top one is the example. The retrieved motions are sorted by similarity from top to bottom.

The calculation overhead of elastic match using dynamic programming is $O(mn)$, where m and n are the lengths of the two compared key-frame sequences, respectively. When retrieving a similar motion, the example is firstly classified into the promising sub-library. The calculation overhead for the classification is $O(hsmn)$, where h is the height of the index tree and s is the size of the sample set in each non-leaf node. After classification, only the similarities between the example and motions from the very sub-library are needed to be calculated and thus the efficiency is well improved.

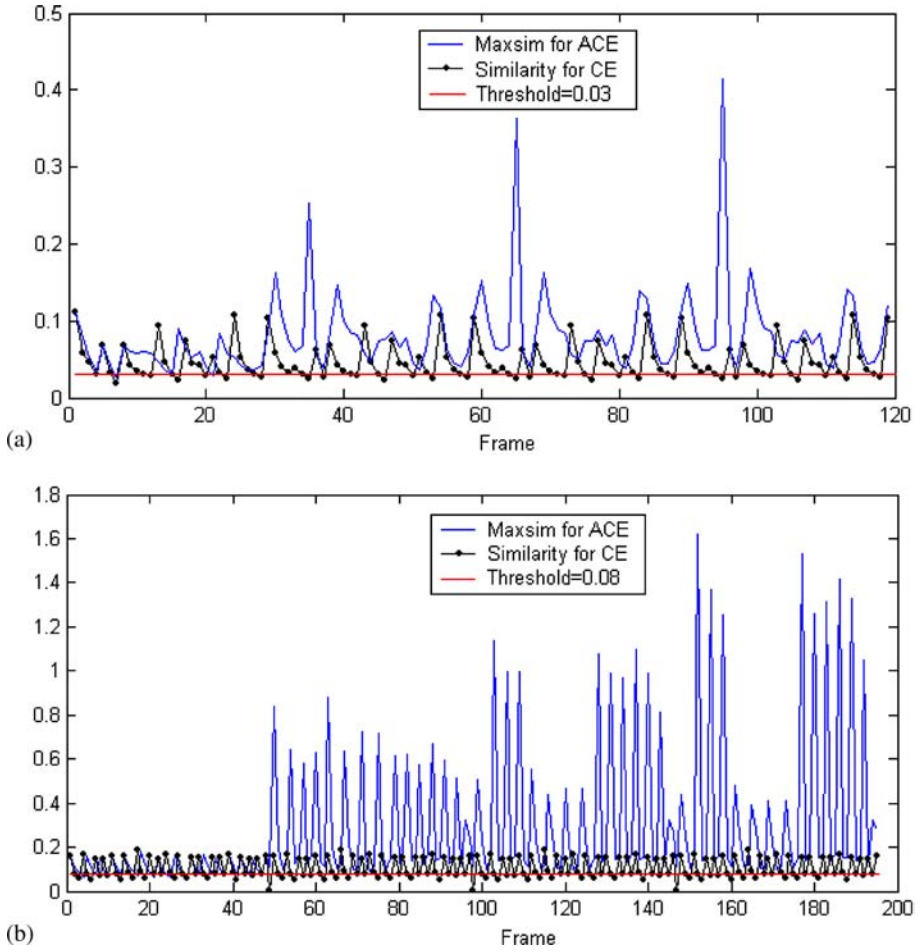

Fig. 6. Extract key-frame sequences from periodical motions using ACE and CE (a) *Power-walk* (b) *Hurdle*.

Table 2
Distribution of examples for queries

Motion class	The number of examples	Motion class	The number of examples
Walk forward	6	Rotate	3
Run forward	5	Run backward	3
Walk backward	5	Climb	2
Jump	4	Box	2
Squat	4	Wave hand	1
Ballet dancing	3	Fall	1

The number of examples from each class is nearly proportional to the size of this class.

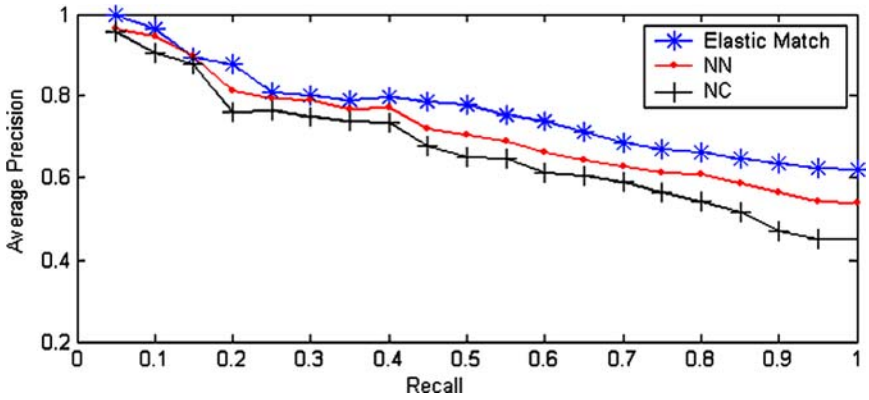

Fig. 7. Performance of elastic match.

Fig. 8. Retrieval result of *walk-forward*.

7. Conclusion

In this paper, a content-based 3D motion retrieval algorithm is proposed. The main contribution is the motion index tree constructed based on the hierarchical motion description. This motion index tree features the hierarchical effect that each joint has on the full-body motion, and serves as a classifier to determine a sub-library that contains promising similar motions to the query example. Thus the efficiency can be well improved. We adopt Nearest Neighbor rule-based dynamic clustering algorithm to partition the motion library and construct the motion index tree. We employ a

novel elastic match algorithm to calculate the similarity between two motions. The elastic match algorithm combines the dynamic time warping and dynamic programming, and has excellent performance on comparing two sequences. To improve the efficiency, we adopt an adaptive clustering-based key-frame extraction algorithm, which is especially competent for key-frame extraction from periodical motions.

The presented work can be used to retrieve similar motions to an example from a motion library. It can also be used in various animation techniques, such as motion editing, analysis, and synthesis. In these promising techniques, the ability to find similar motions is demanded for synthesizing new motions. Moreover, a potential application is to find and provide appropriate motions for autonomous animated characters.

Contrary to such information as image, video, and audio, it is common for animators, the most possible users of 3D motion, to sketch a simple key-frame motion. Compared with the query with a real motion as the example, the query with a scripted motion is usually more relevant to the user's intent. In addition, it is often difficult to obtain an appropriate example. So such user interface as allowing users to script key-frame motions is needed and a corresponding algorithm to compare the scripted example with captured motion in the library is demanded. A promising method is to decompose the captured motion into different bands with wavelet analysis and compare the approximate content with the scripted one. In this paper, we determine the weights for the joints at each level empirically. A reasonable method is needed to calculate the weights automatically. We will address these two questions in the future work.

Acknowledgments

We would like to thank the CVIU editor and reviewers for their constructive comments, Jun Yang for his help to polish the presentation. This work is supported by 973 Program (No. 2002CB312101), the National Natural Science Foundation of China (No. 60272031), Zhejiang Provincial Natural Science Foundation of China (ZD0212), and Doctorate Research Foundation of the State Education Commission of China (No. 20010335049).

References

[1] M. Gleicher, Animation from observation: motion capture and motion editing, Comput. Graph. 4 (33) (1999) 51–55.
[2] K. Pullen, Christoph Bregler, Motion capture assisted animation: texturing and synthesis, in: Proceedings, SIGGRAPH 2002, San Antonio, Texas, 2002, pp. 501–508.
[3] M. Gleicher, Motion editing with spacetime constraints, in: Proceedings, 1997 Symposium on Interactive 3D Graphics, Providence, RI, 1997, pp.139–148.
[4] M. Gleicher, P. Litwinowicz, Constraint-based motion adaptation, J. Visual. Comput. Animat. 9 (2) (1998) 65–94.
[5] Jehee Lee, Sung Yong Shin, A hierarchical approach to interactive motion editing for human-like figures, in: Proceedings, SIGGRAPH 99, Los Angeles, CA, 1999, pp. 39–48.

[6] M. Gleicher, Retargeting motion to new characters, in Proceedings, SIGGRAPH 98, Orlando, FL, 1998, pp. 33–42.
[7] A. Bruderlin, L. Williams, Motion signal processing, in: Proceedings, SIGGRAPH 95, Addison-Wesley, Reading, MA, 1995, pp. 97–104.
[8] Feng Liu, Yueting Zhuang, Zhongxiang Luo, Yunhe Pan, A hybrid motion data manipulation: wavelet based motion processing and spacetime rectification, in: Proceedings, IEEE Pacific Rim Conference on Multimedia 2002, Hsinchu, Taiwan, 2002, pp. 743–750.
[9] Jehee Lee, Jinxiang Chai, Paul S.A. Reitsma, Jessica K. Hodgins, Nancy S. Pollard, Interactive control of avatars animated with human motion data, in: Proceedings, SIGGRAPH 2002, San Antonio, Texas, 2002, pp. 491–500.
[10] Yan Li, Tianshu Wang, Heung-Yeung Shum, Motion texture: a two-level statistical model for character synthesis, in: Proceedings, SIGGRAPH 2002, San Antonio, Texas, 2002, pp. 465–472.
[11] L. Kovar, M. Gleicher, F. Pighin, Motion graphs, in: Proceedings, SIGGRAPH 2002, San Antonio, Texas, 2002, pp. 473–482.
[12] Lei Zhu, Aidong Zhang, Aibing Rao, Rohini Srihari, Keyblock: an approach for content-based image retrieval, in: Proceedings, ACM Multimedia 2000, Los Angeles, CA, 2000, pp. 157–166.
[13] E. Wold, T. Blum, D. Keislar, J. Wheaton, Content-based classification, search and retrieval of audio, IEEE Multimedia Mag. 3 (3) (1996) 27–36.
[14] M. Flickner, H.S. Sawhney, J. Ashley, Qian Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker, Query by image and video content: the QBIC system, IEEE Comput. 28 (9) (1995) 23–32.
[15] Hongjiang Zhang, Jianhua Wu, Di Zhong, S.W. Smoliar, An integrated system for content-based video retrieval and browsing, Pattern Recogn. 30 (4) (1997) 643–658.
[16] M. Mills, J. Cohen, Y.Y. Wong, A magnifier tool for video data, in: Proceedings, ACM Human Computer Interface, Monterey, CA,1992, pp. 93–98.
[17] Y. Taniguchi, A. Akutsu, Y. Tonomura, H. Hamada, An intuitive and efficient access interface to real-time incoming video based on automatic indexing, in: Proceedings, ACM Multimedia, San Francisco, CA, 1995, pp. 25–33.
[18] W. Wolf, Key frame selection by motion analysis, in: Proceedings, ICASSP'96, 1996, vol. 2, pp. 1228–1231.
[19] T. Candemir, L. Shih-Ping, Automatic key-frame selection for content-based video indexing and access, Proc. SPIE, 3972 (2000) 554–563.
[20] Hong Jiang Zhang, Chien Yong Low, Stephen W. Smoliar, Di Zhong, Video parsing, retrieval and browsing: an integrated and content-based solution, in: Proceedings, ACM Multimedia, San Francisco, CA, 1995, pp. 15–24.
[21] Changick Kim, Jenq-Neng Hwang, Object-based video abstraction for video surveillance systems, IEEE Trans. Circuits Syst. Video Technol. 12 (12) (2002) 1128–1138.
[22] Y.S. Avrithis, A.D. Doulamis, N.D. Doulamis, S.D. Kollias, A stochastic framework for optimal key frame extraction from MPEG video databases, CVIU 75 (1999) 3–24.
[23] Michael G. Christel, Michael A. Smith, C. Roy Taylor, David B. Winkler, Evolving video skims into useful multimedia abstractions, in: Proceedings, ACM CHI'98 Conference on Human Factors in Computing Systems, Los Angeles, CA, 1998, pp. 171–178.
[24] ChangIck Kim, Jenq-Neng Hwang, Object-based video abstraction using cluster analysis, in: Proceedings, ICIP2001, Greece, 2001, pp. 657–660.
[25] A.M. Ferman, B. Gunsel, A.M. Tekalp, Object-based indexing of MPEG-4 compressed video, in: Proceedings, SPIE-3024, San Jose, CA, February 1997, pp. 953–963.
[26] B. Erol, F. Kossentini, Automatic key video object plane selection using the shape information in the MPEG-4 compressed domain, IEEE Trans. Multimedia 2 (2000) 129–138.
[27] Chang Ick Kim, Jenq-Neng Hwang, An integrated scheme for object-based video abstraction, in: Proceedings, ACM Multimedia 2000, Los Angeles, CA, 2000, pp. 303–311.
[28] Changick Kim, Jenq-Neng Hwang, Fast and robust moving object segmentation in video sequences, in: Proceedings, ICIP'99, Kobe, Japan, 1999, pp. 131–134.

[29] Changick Kim, Jenq-Neng Hwang, Fast and automatic video object segmentation and tracking for content-based applications, IEEE Trans. Circuits Syst. Video Technol. 12 (2002) 122–129.
[30] Ratakonda, Krishna, Sezan, M. Ibrahim, Crinon J. Regis, Hierarchical video summarization, Proc. SPIE 3653 (1999) 1531–1541.
[31] A.D. Doulamis, N.D. Doulamis, S.D. Kollias, A fuzzy video content representation for video summarization and content-based retrieval, Signal Process. 80 (6) (2000) 1049–1067.
[32] F. Sebastian Grassia, Motion editing: mathematical foundations, in course: Motion Editing: Principles, Practice, and Promise, SIGGRAPH 2000, New Orleans, Louisiana, USA, 2000.
[33] Zhaoqi Bian, Xuegong Zhang, Pattern recognition, Tsinghua University Press, Beijing, 1999, pp. 241–244.
[34] Yueting Zhuang, Yong Rui, Thomas S. Huang, Adaptive key frame extraction using unsupervised clustering, in: Proceedings, IEEE ICIP'98, Chicago, USA, 1998, pp. 886–870.
[35] Li Zhao, Wei Qi, Stan Z. Li, Shiqiang Yang, Hongjiang Zhang, Key-frame extraction and shot retrieval using nearest feature line (NFL), in: Proceedings, International Workshop on Multimedia Information Retrieval, in conjunction with ACM Multimedia Conference 2000, Los Angeles, 2000, pp. 217–220.
[36] R. Martens, L. Claesen, Dynamic programming optimization for online signature verification, in: Proceedings, International Conference on Document Analysis and Recognition, Ulm, Germany, 1997, pp. 653–656.
[37] C.C. Tappert, Adaptive on-line handwriting recognition, in: Proceedings, International Conference on Pattern Recognition, Montreal, Canada, 1984, pp. 1004–1007.

Statement of ownership, management, and circulation required by the Act of October 23, 1962, Section 4369, Title 39, United States Code: of

COMPUTER VISION AND IMAGE UNDERSTANDING

Published monthly by Elsevier Inc., 6277 Sea Harbor Drive, Orlando, FL 32887-4900. Number of issues published annually: 12. Editor: A.C. Kak, Robot Vision Lab, Purdue University, West Lafayette, IN 47907, USA.

Owned by Elsevier Inc., 525 B Street, Suite 1900, San Diego, CA 92101-4495. Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, and other securities: None.
Paragraphs 2 and 3 include, in cases where the stockholder or security holder appears upon the books of the company as trustee or in any other fiduciary relation, the name of the person or corporation for whom such trustee is acting, also the statements in the two paragraphs show the affiant's full knowledge and belief as to the circumstances and conditions under which stockholders and security holders who do not appear upon the books of the company as trustees, hold stock and securities in a capacity other than that of a bona fide owner. Names and addresses of individuals who are stockholders of a corporation which itself is a stockholder or holder of bonds, mortgages, or other securities of the publishing corporation have been included in paragraphs 2 and 3 when the interests of such individuals are equivalent to 1 percent or more of the total amount of the stock or securities of the publishing corporation.
Total no. copies printed: average no. copies each issue during preceding 12 months: 1000; single issue nearest to filing date: 1000. Paid circulation (a) to term subscribers by mail, carrier delivery, or by other means: average no. copies each issue during preceding 12 months: 57; single issue nearest to filing date: 46. (b) Sales through agents, news dealers, or otherwise: average no. copies each issue during preceding 12 months: 619; single issue nearest to filing date: 569. Free distribution (a) by mail: average no. copies each issue during preceding 12 months: 66; single issue nearest to filing date: 62. (b) Outside the mail: average no. copies each issue during preceding 12 months: 9; single issue nearest to filing date: 9. Total no. of copies distributed: average no. copies each issue during preceding 12 months: 741; single issue nearest to filing date: 677. Percent paid and/or requested circulation: average percent each issue during preceding 12 months: 91.2%; single issue nearest to filing date: 90.8%.

(Signed) Paul Richmond, Manufacturing and Operations Manager