Efficient Content-Based Retrieval of Motion Capture Data

Meinard Müller Tido Röder Michael Clausen
University of Bonn, Department of Computer Science III*

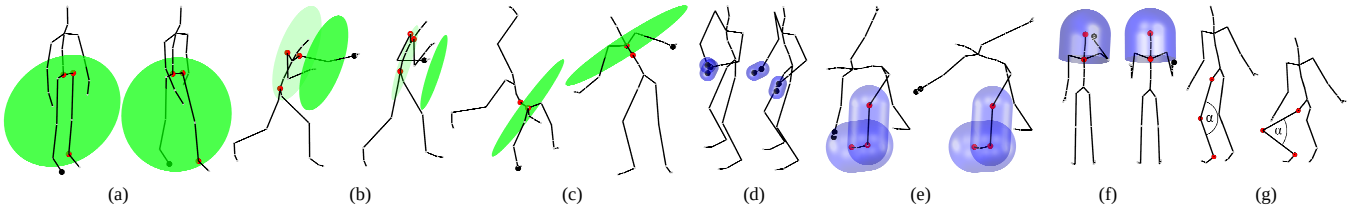

Figure 1: Qualitative features describing geometric relations between the body points of a pose that are indicated by red and black markers.

Abstract

The reuse of human motion capture data to create new, realistic motions by applying morphing and blending techniques has become an important issue in computer animation. This requires the identification and extraction of logically related motions scattered within some data set. Such content-based retrieval of motion capture data, which is the topic of this paper, constitutes a difficult and time-consuming problem due to significant spatio-temporal variations between logically related motions. In our approach, we introduce various kinds of qualitative features describing geometric relations between specified body points of a pose and show how these features induce a time segmentation of motion capture data streams. By incorporating spatio-temporal invariance into the geometric features and adaptive segments, we are able to adopt efficient indexing methods allowing for flexible and efficient content-based retrieval and browsing in huge motion capture databases. Furthermore, we obtain an efficient preprocessing method substantially accelerating the cost-intensive classical dynamic time warping techniques for the time alignment of logically similar motion data streams. We present experimental results on a test data set of more than one million frames, corresponding to 180 minutes of motion. The linearity of our indexing algorithms guarantees the scalability of our results to much larger data sets.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; H.3 [Information Storage and Retrieval]: Information Search and Retrieval

Keywords: motion capture, geometric feature, adaptive segmentation, indexing, retrieval, time alignment

1 Introduction

The generation of human motion capture data as used in data-driven computer animations is a time-consuming and expensive process. Various editing and morphing techniques for the modification and adaptation of existing motion data [Bruderlin and Williams 1995; Witkin and Popovic 1995] or for the synthesis of new, realistic motions from example motions [Giese and Poggio 2000; Pullen and Bregler 2002; Kovar and Gleicher 2003] have been developed in the last few years. Prior to reusing and processing motion capture material, one has to solve the fundamental problem of *identifying* and *extracting* suitable motion clips from the database on hand. In doing so, a user may describe the motion clips to be retrieved in various ways at different semantic levels. One possible specification could be a rough textual description such as "a kick of the right foot followed by a punch." Another query mode would involve a short query motion clip, the task being to retrieve all clips in the database containing parts or aspects similar to the query. This kind of problem is commonly referred to as *content-based retrieval*. In this paper, we present a prototypical system for content-based motion retrieval, where the query consists of a motion clip as well as a user-specified selection of motion aspects to be considered in the retrieval process. Unlike the work of Kovar and Gleicher [2004], our technique does not support fully automatic "query-by-example," since the user has to supply additional query-dependent input. Being able to choose certain motion aspects, however, provides the user with a high degree of flexibility, cf. the subsequent overview.

The crucial point in content-based motion retrieval is the notion of "similarity" used to compare different motions. Intuitively, two motions may be regarded as similar if they represent variations of the same action or sequence of actions, see Kovar and Gleicher [2004]. Here the variations may concern the spatial as well as the temporal domain. For example, the two walking motions shown in Fig. 5 and Fig. 6, respectively, may be perceived as similar even though they differ considerably in their respective speeds. In other words, *logically similar* motions need not be *numerically similar*, as is also pointed out by Kovar and Gleicher [2004]. This may lead to incomplete and dissatisfying retrieval results when using similarity measures based on numerical comparison of spatial coordinates. Furthermore, the necessary warping of the time axis to establish frame correspondences is computationally expensive, making this kind of technique infeasible for large data sets, see also Sect. 2.

To bridge the semantic gap between logical similarity as perceived by humans and computable numerical similarity measures, we introduce new types of qualitative geometric features and induced motion segmentations, yielding spatio-temporal invariance as needed to compare logically similar motions. This strategy has far-reaching consequences regarding efficiency, flexibility and automation in view of indexing, content-based retrieval and time alignment of motion capture data. The following overview summarizes the main contributions of this paper.

*e-mail: {meinard, roedert, clausen}@cs.uni-bonn.de

1.1 Overview

1. **Geometric Features:** We introduce a class of boolean features expressing geometric relations between certain body points of a pose. As an example of this kind of features, consider the test whether the right foot lies in front of or behind the plane spanned by the left foot, the left hip joint and the center of the hip (the root), cf. Fig. 1 (a). Such geometric features are very robust to spatial variations and allow the identification of logically corresponding events in similar motions. In particular, (user-specified) combinations of such qualitative features become a powerful tool in describing and specifying motions at a high semantic level, see also Fig. 2.

2. **Adaptive Segmentation:** In conventional approaches, feature extraction is often performed in two steps: first, the data stream is segmented along the time axis, then a feature vector is computed for each of the resulting segments. We suggest a different approach: for a fixed combination of pose-based geometric features, consider consecutive frames yielding the same feature vectors, in the following simply referred to as *runs*. The *segments* of a given motion data stream are then defined to be runs of maximal length. Such segments not only inherit the semantic qualities of the underlying features but are also robust to the local time variations that are typical of logically related motions. Furthermore, changing the combination of features will automatically lead to an adaptation of the induced segmentation. As an example, see Figs. 3 and 5.

3. **Similarity and Indexing:** In plain words, our method coarsens each motion data stream by transforming it into a sequence of geometric configurations. Two motion clips are then considered as similar if they possess (more or less) the same progression of geometric features. Opposed to recent approaches that involve dynamic time warping (DTW) to establish correspondence of related events, our approach incorporates spatio-temporal invariance in the geometric features and induced segments. This allows us to employ standard information retrieval techniques for fast content-based and fault tolerant retrieval based on indexing with inverted lists. We simply use the feature vectors as index words, and indexing is carried out at the segment level rather than at the frame level. This leads to significant savings in memory and running time. In particular, the time and space required to build and store our index structure is *linear*, $O(n)$, in the number n of database frames opposed to DTW-based strategies, which are *quadratic*, $O(n^2)$, in n, see Sect. 6.1.

4. **Queries and Retrieval:** In our system, a query consists of a short motion clip and a query-dependent specification of motion aspects that determines the desired notion of similarity. For the latter, the user selects relevant features from a given set of intuitive, programmer-defined geometric features (each expressing a relation of certain body parts). On the one hand, this does not allow for fully automatic (query-independent) motion retrieval, as would be necessary to process a whole batch of motion queries. On the other hand, feature selection enables the user to incorporate his previous knowledge of the query motion into the retrieval process. Thus, the user may select or mask out certain aspects such as restricted body areas in the query, so that, for example, all instances of "clapping ones hands" can be found irrespective of any concurrent locomotion. Furthermore, many movements such as "kicking with subsequent punch" or "standing up and clapping ones hands" can be specified by a short sequence of key poses, which translate into a typical progression of geometric constellations. Therefore, our approach is a major step towards handling such low-level descriptive queries in an automatic way without using manually generated annotations.

5. **Time Alignment:** Matching two feature progressions obtained from similar motions can be regarded as a time alignment of the underlying motion data streams. This fact can be used to significantly accelerate classical DTW-based alignment procedures by first computing (in linear time) a coarse match based on geometric features and then refining this alignment with classical DTW techniques.

1.2 Notations

For the sake of clarity, we quickly introduce some notions used in the rest of this paper. Human motion is commonly modeled using a *kinematic chain*, which may be thought of as a simplified copy of the human skeleton. A kinematic chain consists of *body segments* (the bones) that are connected by *joints* of various types. In the following, let J denote the set of joints, where each joint is referenced by an intuitive term such as 'root', 'lankle' (for 'left ankle'), 'rankle' (for 'right ankle'), 'lknee' (for 'left knee'), and so on. For simplicity, end effectors such as toes or fingers are also regarded as joints. Using motion capture techniques, one can derive from an actor's motion a time-dependent sequence of 3D joint coordinates as well as joint angles with respect to some fixed kinematic chain. In the following, a *motion capture data stream* is thought of as a sequence of *frames*, each frame specifying the 3D coordinates of the joints at a certain point in time. Moving from the technical background to an abstract geometric context, we also speak of a *pose* instead of a frame. Mathematically, a pose can be regarded as a matrix $P \in \mathbb{R}^{3\times|J|}$, where $|J|$ denotes the number of joints. The j-th column of P, denoted by P^j, corresponds to the 3D coordinates of joint $j \in J$. A motion capture data stream (in information retrieval terminology also referred to as a *document*) can be modeled as a function $D : [1:T] \to \mathscr{P} \subset \mathbb{R}^{3\times|J|}$, where T denotes the number of poses, $[1:T] := \{1,2,\ldots,T\}$ corresponds to the time axis (for a fixed sampling rate), and $\mathscr{P}$ denotes the set of poses. A subsequence of consecutive frames is also referred to as a *motion clip*. Finally, the curve described by the 3D coordinates of a single body point is termed *trajectory*.

2 Related Work

In view of massively growing multimedia databases of various types and formats, efficient methods for indexing and content-based retrieval have become an important issue. Vast literature exists on indexing and retrieval in text, image, and video data, see, e.g., Witten et al. and Bakker et al. [1999; 2003] and references therein. For the music scenario, Clausen and Kurth [2004] give a unified approach to content-based retrieval; their group theoretical concepts generalize to other domains as well. The problem of indexing large time series databases has also attracted great interest in the database community, see, e.g., Last et al. and Keogh [2004; 2002] and references therein. Here, possible distortions of the time axis constitute a major problem in comparing related time series, usually solved by means of dynamic time warping. DTW, however, is cost-intensive in computing time and memory. Keogh [2002] describes an indexing method based on lower bounding techniques that makes retrieval of time-warped time series feasible even for large data sets.

Only recently, motion capture data has become publicly available on a larger scale (e.g., CMU [2003]), reinforcing the demand for efficient indexing and retrieval methods. Such methods are necessary to efficiently retrieve logically related motions, which can

then be processed via editing and morphing techniques [Bruderlin and Williams 1995; Witkin and Popovic 1995; Giese and Poggio 2000; Pullen and Bregler 2002; Kovar and Gleicher 2003; Kovar and Gleicher 2004]. So far, only little work has been published on motion capture indexing and content-based motion retrieval. We give a short overview of the relevant literature: to identify motions similar to a given query motion, Wu et al. [2003] proceed in two stages: they first identify start and end frames of possible candidate clips utilizing a pose-based index and then compute the actual distance from the query via DTW. Cardle et al. [2003] sketch how to use DTW-based indexing techniques to accomplish the retrieval task. Adapting techniques from Keogh [2002], Keogh et al. [2004] describe how to efficiently identify similar motion fragments that differ by some uniform scaling factor with respect to the time axis. In the approach of Kovar and Gleicher [2004], numerically similar motions are identified by means of a DTW-based index structure termed *match web*. A multi-step search spawning new queries from previously retrieved motions allows for the identification of logically similar motions using numerically similar motions as intermediaries. The authors report good retrieval results, which are particularly suitable for their blending application. Our approach to indexing and content-based retrieval of motion capture data differs fundamentally from previous approaches: opposed to DTW-based techniques that rely on suitable numerical local cost-measures to compare individual frames of data streams, we grasp spatio-temporal invariance in our features and induced segments, allowing for exact matchings at the segment level.

The idea of considering geometric (combinatorial, relational, qualitative) features instead of numerical (metrical, quantitative) features is not new and has already been applied by, e.g., Carlsson [1996; 1999] as well as Sullivan and Carlsson [2002] in other domains such as visual object recognition in 2D and 3D, or action recognition and tracking. The following observations are of fundamental importance, see Carlsson [1996]: firstly, relational structures are not only interesting for general recognition problems (due to their invariance properties) but also ideally suited for indexing (due to their discrete nature). Secondly, *relational* similarity of shapes correlates quite well with *perceptual (logical)* similarity. These principles motivate the usage of progressions of geometric constellations in order to identify logically similar movements. Analogously, Green and Guan [2004] use progressions of kinematic features such as motion vectors of selected joints, so-called *dynemes*, as basic building blocks of their HMM-based motion models.

3 Geometric Features

In this section, we define various kinds of *geometric features* describing geometric relations between specified points of the kinematic chain for some fixed, isolated pose. To this end, we need the notion of a *boolean feature*, which we describe mathematically as a boolean function $F : \mathscr{P} \to \{0,1\}$. Obviously, any boolean expression of boolean functions (evaluated pose-wise) is a boolean function itself, examples being the conjunction $F_1 \wedge F_2$ and the disjunction $F_1 \vee F_2$ of boolean functions F_1 and F_2. Forming a vector of f boolean functions for some $f \geq 1$, one obtains a combined function $F : \mathscr{P} \to \{0,1\}^f$. From this point forward, F will be referred to as a *feature function* and the vector $F(P)$ as a *feature vector* or simply a *feature* of the pose $P \in \mathscr{P}$. Any feature function can be applied to a motion capture data stream $D : [1:T] \to \mathscr{P}$ in a pose-wise fashion, which is expressed by the composition $F \circ D$.

We now consider a special class of geometrically meaningful feature functions. As an example, a geometric feature may express whether the right toes lie in front of the plane spanned by the left ankle, the left hip and the root for a fixed pose. More generally, let $p_i \in \mathbb{R}^3$, $1 \leq i \leq 4$, be four 3D points, the first three of which are in general position. Let $\langle p_1, p_2, p_3 \rangle$ denote the oriented plane spanned by the first three points, where the orientation is determined by point order. Then define

$$B(p_1,p_2,p_3;p_4) := \begin{cases} 1, & \text{if } p_4 \text{ lies in front of or on } \langle p_1,p_2,p_3\rangle, \\ 0, & \text{if } p_4 \text{ lies behind } \langle p_1,p_2,p_3\rangle. \end{cases} \tag{1}$$

From this we obtain a feature function $F_{\text{plane}}^{(j_1,j_2,j_3;j_4)} : \mathscr{P} \to \{0,1\}$ for any four distinct joints $j_i \in J$, $1 \leq i \leq 4$, by defining

$$F_{\text{plane}}^{(j_1,j_2,j_3;j_4)}(P) := B(P^{j_1},P^{j_2},P^{j_3};P^{j_4}). \tag{2}$$

The concept of such geometric features is simple but powerful, as we will illustrate by continuing the above example. Setting $j_1 =$‘root’, $j_2 =$‘lankle’, $j_3 =$‘lhip’, and $j_4 =$‘rtoes’, we denote the resulting feature by $F^r := F_{\text{plane}}^{(j_1,j_2,j_3;j_4)}$. The plane determined by j_1, j_2, and j_3 is indicated in Fig. 1 (a) as a green disc. Obviously, the feature $F^r(P)$ is 1 for a pose P corresponding to a person standing upright. It assumes the value 0 when the right foot moves to the back or the left foot to the front, which is typical for locomotion such as walking or running. Interchanging corresponding left and right joints in the definition of F^r and flipping the orientation of the resulting plane, we obtain another feature function denoted by F^l. Let us have a closer look at the feature function $F := F^r \wedge F^l$, which is 1 if and only if both, the right as well as the left toes, are in front of the respective planes. It turns out that F is very well suited to characterize any kind of walking or running movement. If a data stream $D : [1:T] \to \mathscr{P}$ describes such a locomotion, then $F \circ D$ exhibits exactly two peaks for any locomotion cycle, from which one can easily read off the speed of the motion (see Fig. 2). On the other hand, the feature F is invariant under global orientation and position, the size of the skeleton, and various local spatial deviations such as sideways and vertical movements of the legs. Of course, F leaves any upper body movements unconsidered.

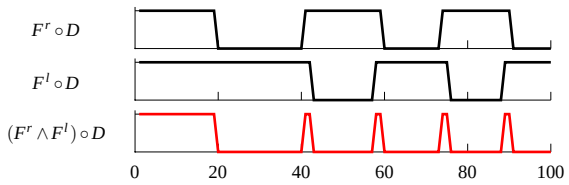

Figure 2: Boolean features F^r, F^l, and the conjunction $F^r \wedge F^l$ applied to the 100-frame walking motion $D = D_{\text{walk}}$ of Fig. 3.

In general, feature functions defined purely in terms of geometric entities that are expressible by joint coordinates are invariant under global transforms such as Euclidean motions and scalings. Geometric features are very coarse in the sense that they express only a single geometric aspect, masking out all other aspects of the respective pose. This makes such features robust to variations in the motion capture data stream that are not correlated with the aspect of interest. Using suitable boolean expressions and combinations of several geometric features then allows to focus on or to mask out certain aspects of the respective motion, see Sect. 5.

The four joints in $F_{\text{plane}}^{(j_1,j_2,j_3;j_4)}$ can be picked in various meaningful ways. For example, in the case $j_1 =$‘root’, $j_2 =$‘lshoulder’, $j_3 =$‘rshoulder’, and $j_4 =$‘lwrist’, the feature expresses whether the left hand is in front of or behind the body. Introducing suitable offsets, one can make the feature more robust. For the previous example, we moved the plane $\langle P^{j_1},P^{j_2},P^{j_3}\rangle$ to the front by one length of the skeleton’s humerus. One thus obtains a more robust feature

Feature set	Features and corresponding abbreviations
F_ℓ	right/left foot in front (F_ℓ^1/F_ℓ^2), right/left foot raised (F_ℓ^3/F_ℓ^4), right/left foot fast (F_ℓ^5/F_ℓ^6), right/left knee bent (F_ℓ^7/F_ℓ^8), right/left leg sideways (F_ℓ^9/F_ℓ^{10}), legs crossed (F_ℓ^{11})
F_{u}	right/left hand in front ($F_{\mathrm{u}}^1/F_{\mathrm{u}}^2$), right/left hand raised ($F_{\mathrm{u}}^3/F_{\mathrm{u}}^4$), right/left arm sideways ($F_{\mathrm{u}}^5/F_{\mathrm{u}}^6$), right/left elbow bent ($F_{\mathrm{u}}^7/F_{\mathrm{u}}^8$), right/left hand fast ($F_{\mathrm{u}}^9/F_{\mathrm{u}}^{10}$), arms crossed ($F_{\mathrm{u}}^{11}$), hands touching ($F_{\mathrm{u}}^{12}$)
F_{m}	right/left hand touching any leg ($F_{\mathrm{m}}^1/F_{\mathrm{m}}^2$), right/left hand touching head or neck ($F_{\mathrm{m}}^3/F_{\mathrm{m}}^4$), right/left hand touching hip area ($F_{\mathrm{m}}^5/F_{\mathrm{m}}^6$), torso bent ($F_{\mathrm{m}}^7$), root fast ($F_{\mathrm{m}}^8$)

Table 1: The three sets of features used in our experiments. F_ℓ characterizes the lower body, F_{u} the upper body and F_{m} the interaction of upper and lower body.

that can distinguish between a pose with a hand reaching out to the front and a pose with a hand kept close to the body, see Fig. 1 (b).

We sketch some other kinds of geometric features that turned out to be useful in our experiments. Instead of specifying the plane by three joints, one can define the plane by a normal vector given by two joints. For example, using the plane that is normal to the vector from the joint 'chest' to the joint 'neck', one can easily check whether a hand is raised above neck height or not, cf. Fig. 1 (c). Another type of expressive geometric features checks whether two joints, two body segments, or a joint and a body segment are within an ε-distance of each other or not. Here one may think of situations such as two hands touching each other, or a hand touching the head or a leg, see Fig. 1 (d)–(f). In our experiments, we chose a relatively large threshold ε, opting for possible false positives rather than having false dismissals. We also use geometric features to check whether certain parts of the body such as the arms, the legs, or the torso are bent or stretched. This is done by measuring the angle α between suitable body segments such as the thigh and the lower leg (corresponding to the knee angle), the upper arm and the forearm (corresponding to the elbow angle), or the spine and the left and right thigh (taking the maximum of the two angles). Our experiments showed that an angle of 120 degrees is a good threshold to classify the respective body part as bent ($\alpha < 120$) or stretched ($\alpha \geq 120$), see Fig. 1 (g). Finally, we complemented our set of geometric features with a couple of non-geometric boolean features accounting for parameters such as absolute speed of certain joints, or relative speed of certain joints with respect to other joints.

In our retrieval scenario, we supply a fixed set of semantically rich features from which the user may select relevant features for the query specification. Concerning feature design, one has to deal with the problem that the feature set should be extensive enough to characterize a broad spectrum of different motion types. Simultaneously, the features should be pairwise orthogonal without over-specializing on certain types of motions, as well as small in number in view of efficiency. For the time being, we chose our features manually, possibly yielding boolean features that manage to capture important motion characteristics. An automated way of composing a high-quality standard feature set backed up by physiological studies of human motions would be an important issue to be considered in the future.

In particular, we designed 31 boolean features divided into three sets F_ℓ, F_{u}, and F_{m}, see Table 1. The features in F_ℓ and F_{u} express properties of the lower/upper part of the body (mainly of the legs/arms, respectively), whereas the features in F_{m} mainly express interactions of the upper and lower part. Here, the idea was to subdivide the space of possible end effector locations into a small set of pose-dependent space "octants" defined by three intersecting planes each (above/below, left/right, in front of/behind). Even though these 31 features are not capable of covering all aspects of all kinds of motions (e.g., motions that are small-scaled enough to take place exclusively within a single space octant), they are sufficient for our prototypical retrieval system. Here, our goal is to retrieve motions based on their rough outlines. To handle queries at a higher level of detail, more refined sets of features are required.

4 Adaptive Segmentation

As mentioned in the introduction, two logically similar motions may exhibit considerable spatial as well as temporal deviations. The pose-based geometric features defined in the last section are invariant under such spatial variations. Introducing the concept of adaptive segmentation, we show in this section how to achieve invariance under local time deformations.

Let F be a feature function. We say that two poses $P_1, P_2 \in \mathscr{P}$ are *F-equivalent* if the corresponding feature vectors $F(P_1)$ and $F(P_2)$ coincide, i.e., $F(P_1) = F(P_2)$. Then, an *F-run* of D is defined to be a subsequence of D consisting of consecutive F-equivalent poses, and the *F-segments* of D are defined to be the F-runs of maximal length. We illustrate these definitions by continuing the example from Sect. 3. Let $F^2 := (F^r, F^l) : \mathscr{P} \to \{0,1\}^2$ be the combined feature formed by F^r and F^l so that the pose set $\mathscr{P}$ is partitioned into four F^2-equivalence classes. Applying F^2 to the walking motion D_{walk} (see also Fig. 2) results in the segmentation shown in Fig. 3, where the trajectories of selected joints have been plotted. F^2-equivalent poses are indicated by the same trajectory color: the color *red* represents the feature vector $(1,1)$, *blue* the vector $(1,0)$, and *green* the vector $(0,1)$. Note that no pose with feature vector $(0,0)$ appears in D_{walk}. Altogether, there are ten runs of maximal length constituting the F^2-segmentation of D_{walk}.

It is this feature-dependent segmentation that accounts for the postulated temporal invariance, the main idea being that motion capture data streams can now be compared at the segment level rather than at the frame level. To be more precise, let us start with the sequence of F-segments of a motion capture data stream D. Since each segment corresponds to a unique feature vector, the segments induce a sequence of feature vectors, which we simply refer to as the *F-feature sequence* of D and denote by $F[D]$. If K is the number of F-segments of D and if $D(t_k)$ for $t_k \in [1:T]$, $1 \leq k \leq K$, is a pose of the k-th segment, then $F[D] = (F(D(t_1)), F(D(t_2)), \ldots, F(D(t_K)))$. For example, for the data stream D_{walk} and the feature function F^2 from Fig. 3, we obtain

$$F^2[D_{\mathrm{walk}}] = \left(\tbinom{1}{1}, \tbinom{0}{1}, \tbinom{1}{1}, \tbinom{1}{0}, \tbinom{1}{1}, \tbinom{0}{1}, \tbinom{1}{1}, \tbinom{1}{0}, \tbinom{1}{1}, \tbinom{0}{1}\right). \tag{3}$$

Obviously, any two adjacent vectors of the sequence $F[D]$ are distinct. The crucial point is that time invariance is incorporated into the F-segments: two motions that differ by some deformation of the time axis will yield the same F-feature sequences. This fact is illustrated by Figs. 5 and 6. Another property is that the segmentation automatically adapts to the selected features, as a comparison of Fig. 3 and Fig. 5 shows. In general, fine features, i.e., feature functions with many components, induce segmentations with many short segments, whereas coarse features lead to a smaller number of long segments.

The main idea is that two motion capture data streams D_1 and D_2 can now be compared via their F-feature sequences $F[D_1]$ and $F[D_2]$ instead of comparing the data streams on a frame-to-frame basis. This has several consequences: first, since spatial and temporal invariance are incorporated in the features and segments, one can use efficient methods from (fault-tolerant) text retrieval to compare the data streams instead of applying cost-intensive techniques such as DTW at the frame level. Second, the number K of segments

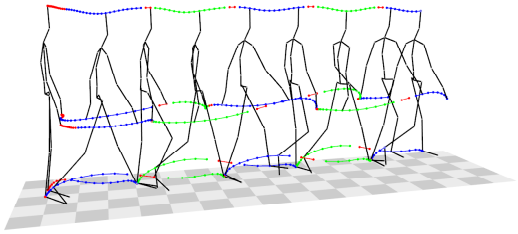

Figure 3: F^2-segmentation of D_{walk}, where F^2-equivalent poses are indicated by uniformly colored trajectory segments. The trajectories of the joints 'head-top','rankle','lankle','rfingers', and 'lfingers' are shown.

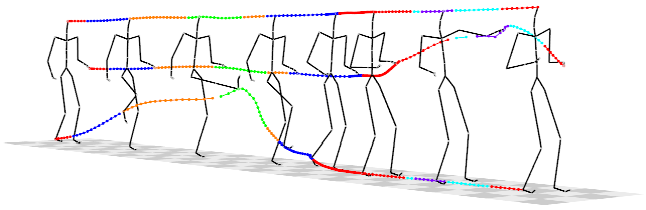

Figure 4: Right foot kick followed by a left hand punch. The segments are induced by a 4-component feature function comprising the thresholded angles for 'rknee' (Fig. 1 (g)) and 'lelbow' as well as features describing the relations "right foot raised" and "left hand in front" (Fig. 1 (b)).

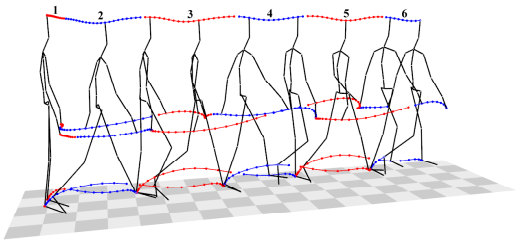

Figure 5: Restricting $F^2 = (F^r, F^l)$ to its first component results in an F^r-segmentation, which is coarser than the F^2-segmentation shown in Fig. 3.

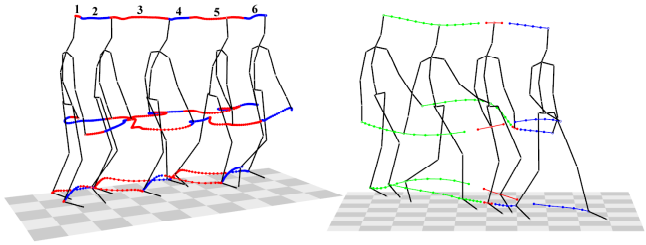

Figure 6: The left part shows five steps of a slow walking motion performed by an elderly person resulting in exactly the same F^r-feature sequence as the much faster motion of Fig. 5. The right part shows a possible query motion resulting in the F^2-feature sequence of the example given in (8).

is generally much smaller than the number T of frames, which accounts for very efficient computations. Finally, the user-specified selection of motion aspects can be realized by considering certain components of $F[D_1]$ and $F[D_2]$ in the comparison, see Sect. 5.

5 Indexing and Retrieval

In the following, we think of a database as a collection $\mathscr{D} = (D_1, D_2, \ldots, D_N)$ of motion capture data streams (documents) D_n, $n \in [1:N]$. To simplify things, we may assume that $\mathscr{D}$ consists of one large document D by concatenating the documents $D_1, \ldots, D_N$, keeping track of document boundaries in a supplemental data structure. Furthermore, we fix a feature function F comprising as components all the features from which the user may select (consisting, e.g., of the 31 features of Table 1). In Sect. 5.1, we present the basic idea of an exact hit with respect to F, formalizing the notion of similarity between motion clips. To account for the user's feature selection as well as uncertainties in the query, we introduce the concept of a fuzzy hit, softening the notion of an exact hit, see Sect. 5.2. To afford an efficient retrieval, exact as well as fuzzy, we once and for all perform a preprocessing step, in which we construct a query-independent index structure for F based on inverted lists. In Sect. 5.3, we explain how to control the number of index words by accepting a small computational and storage overhead. The concepts introduced in this section are well known in other domains, cf. Clausen and Kurth [2004].

5.1 Hits and Index-Based Search

Intuitively, we want to consider two motion clips as similar if they exhibit the same progression of geometric constellations or, in other words, if they have the same feature sequence. An *exact F-hit* is defined to be a segment number $k \in \mathbb{N}$ such that $F[Q]$ is a subsequence of consecutive feature vectors in the F-sequence $F[D]$ starting from the k-th segment. For short, we then write $F[Q] \sqsubset_k F[D]$. In other words, if $F[Q] = (v_1, v_2, \ldots, v_L)$ with $v_\ell \in \{0,1\}^f$ for $1 \le \ell \le L$ and $F[D] = (w_1, w_2, \ldots, w_M)$ with $w_m \in \{0,1\}^f$ for $1 \le m \le M$, then

$$F[Q] \sqsubset_k F[D] \;\Leftrightarrow\; w_k = v_1, w_{k+1} = v_2, \ldots, w_{k+L-1} = v_L. \quad (4)$$

Note that even though this concept of a hit is exact at the feature and segment level, it still permits a lot of spatial and temporal variation at the frame level. The set of all exact F-hits in the database $\mathscr{D}$ is given by

$$H_{\mathscr{D}}(F[Q]) := \{k \in [1:M] \mid F[Q] \sqsubset_k F[D]\}. \quad (5)$$

The important point is that the set $H_{\mathscr{D}}(F[Q])$ can be evaluated very efficiently using *inverted lists*. To this end, we store a list

$$L(v) := H_{\mathscr{D}}((v)) = \{k \in [1:M] \mid (v) \sqsubset_k F[D]\} \quad (6)$$

for each feature vector $v \in \{0,1\}^f$. An element $k \in L(v)$ tells us that the k-th feature vector of $F[D]$ equals v. Let $F[Q] = (v_1, v_2, \ldots, v_L)$ as before; then it is easily verified that

$$H_{\mathscr{D}}(F[Q]) = \bigcap_{\ell \in [1:L]} (L(v_\ell) - \ell + 1), \quad (7)$$

where we set $L(v) - \ell + 1 := \{k - \ell + 1 \mid k \in L(v)\}$. In other words, $H_{\mathscr{D}}(F[Q])$ can be obtained by intersecting suitably shifted inverted lists corresponding to the feature vectors of $F[Q]$. This suggests to build an index in the following manner: compute and store for each feature vector $v \in \{0,1\}^f$ the inverted list $L(v)$, which can be done in a preprocessing step, independently of any query. Keep the entries k within each list $L(v)$ sorted according to the canonical ordering of $\mathbb{Z}$. Then the computation of unions and intersections of the inverted lists can be processed efficiently using merge operations or binary search. The resulting index consisting of all inverted lists $L(v)$ will be denoted by $I_F^{\mathscr{D}}$.

As an example, consider $D = D_{\text{walk}}$ and $F = F^2$ from Fig. 3. Then $L(\binom{1}{1}) = \{1,3,5,7,9\}$, $L(\binom{0}{1}) = \{2,6,10\}$, $L(\binom{1}{0}) = \{4,8\}$,

and $L(\binom{0}{0}) = \emptyset$. Now, assume that for a query Q one obtains the F-feature sequence $F[Q] = (\binom{1}{0},\binom{1}{1},\binom{0}{1})$, then

$$\begin{aligned} H_{\mathscr{D}}(F[Q]) &= L(\tbinom{1}{0}) \cap \big(L(\tbinom{1}{1}) - 1\big) \cap \big(L(\tbinom{0}{1}) - 2\big) \\ &= \{4,8\} \cap \{0,2,4,6,8\} \cap \{0,4,8\} \\ &= \{4,8\}. \end{aligned} \tag{8}$$

In other words, there are two F-hits for Q starting with the fourth and eighth segment of $F[D]$, respectively.

5.2 Fuzzy Search with Adaptive Segmentation

The feature function F is chosen to cover a broad spectrum of aspects appearing in all types of motions. Therefore, considering a specific motion class, many of the features are irrelevant for retrieval and should be masked out to avoid a large number of false negatives due to over-specification. For example, considering locomotion, the user may not be interested in any movement of the arms. Furthermore, the user may be unsure about certain parts of the query and may want to leave them unspecified. To handle such situations, we introduce the concept of *fuzzy search*. Instead of considering a feature sequence $F[Q] = (v_1, v_2, \ldots, v_L)$ of a query Q, we now allow for each v_ℓ a finite set $V_\ell \subseteq \{0,1\}^f$ with $v_\ell \in V_\ell$, which can be thought of as a set of *alternatives* to v_ℓ. Then a *fuzzy query* is defined to be a sequence $F[Q]_{\text{fuz}} := (V_1, V_2, \ldots, V_L)$ of fuzzy sets such that $V_\ell \cap V_{\ell+1} = \emptyset$ for $1 \leq \ell < L$. (In case the latter intersection condition—introduced for technical reasons—is not fulfilled, we iteratively conjoin adjacent sets with nonempty intersection until we end up with a sequence having the desired property. Note that this procedure corresponds to coarsening the segmentation of the query.) Extending the definition of (4), a *fuzzy hit* is an element $k \in [1:M]$ such that $F[Q]_{\text{fuz}} \sqsubset_k F[D]$, where we set

$$F[Q]_{\text{fuz}} \sqsubset_k F[D] \;:\Leftrightarrow\; w_k \in V_1, w_{k+1} \in V_2, \ldots, w_{k+L-1} \in V_L \tag{9}$$

with $F[D] = (w_1, w_2, \ldots, w_M)$ as above. Obviously, the case $V_\ell = \{v_\ell\}$ for $1 \leq \ell \leq L$ reduces to the case of an exact hit, cf. (4). Similar to (5), the set of all fuzzy hits is defined to be

$$H_{\mathscr{D}}(F[Q]_{\text{fuz}}) := \{k \in [1:M] \mid F[Q]_{\text{fuz}} \sqsubset_k F[D]\}. \tag{10}$$

In analogy to (7), the set $H_{\mathscr{D}}(F[Q]_{\text{fuz}})$ can be computed via

$$H_{\mathscr{D}}(F[Q]_{\text{fuz}}) := \bigcap_{\ell \in [1:L]} (L(V_\ell) - \ell + 1) \text{ with } L(V_\ell) := \bigcup_{v \in V_\ell} L(v). \tag{11}$$

Obviously, the admission of alternatives can be realized very efficiently in terms of unions of inverted lists, thus avoiding the $\prod_{\ell=1}^{L} |V_\ell|$ individual queries for all combinations of fuzzy features that (9) might suggest to be necessary.

The fuzzy concept introduced above is not yet exactly what we want: so far, the fuzziness only refers to the spatial domain (allowing alternative choices for the pose-based features) but ignores the temporal domain. More precisely, the segmentation of D is only determined by the feature function, disregarding the fuzziness of the fuzzy query $F[Q]_{\text{fuz}}$. To adjust the temporal segmentation to the fuzziness of the query, we proceed as follows: let $F[D] = (w_1, w_2, \ldots, w_M)$ be the F-feature sequence of D. Supposing $w_k \in V_1$ with $w_{k-1} \notin V_1$ for some index $k_1 := k \in [1:M]$, we determine the maximal index $k_2 \geq k_1$ with $w_m \in V_1$ for all $m = k_1, k_1+1, \ldots, k_2-1$ and concatenate all segments corresponding to these w_m into one large segment. By construction, $w_{k_2} \notin V_1$. Only if $w_{k_2} \in V_2$, we proceed in the same way, determining some maximal index $k_3 > k_2$ with $w_m \in V_2$ for all $m = k_2, k_2+1, \ldots, k_3-1$, and so on. In case we find a sequence of indices $k_1 < k_2 < \ldots < k_L$ constructed iteratively in this fashion, we say that $k \in [1:M]$ is an *adaptive fuzzy hit*.

In other words, when comparing the fuzzy query $F[Q]_{\text{fuz}}$ with a document D, the F-segmentation of D is coarsened to obtain maximal runs of frames with respect to a set V_ℓ of feature vectors rather than to a single feature vector v_ℓ. The above procedure not only checks for the existence of such adaptive fuzzy hits but also constructs the respective adaptive segmentation corresponding to the fuzzy query. Again, one can efficiently compute the set of all adaptive fuzzy hits, denoted by $H_{\mathscr{D}}(F[Q]^{\text{ad}}_{\text{fuz}})$, based on the inverted lists of our index. The details are of rather technical nature and will be published elsewhere.

We illustrate the above procedure by continuing our example $D = D_{\text{walk}}$ and $F = F^2$ from Fig. 3. Let's start with a fuzzy query $F[Q]_{\text{fuz}} = (V_1, V_2, V_3)$ with $V_1 = V_3 = \{\binom{0}{0},\binom{0}{1}\}$ and $V_2 = \{\binom{1}{0},\binom{1}{1}\}$. Looking at the feature sequence $F[D]$ shown in (3), one easily derives that $H_{\mathscr{D}}(F[Q]^{\text{ad}}_{\text{fuz}}) = \{2,6\}$, i.e., there are exactly two adaptive fuzzy hits in D (opposed to $H_{\mathscr{D}}(F[Q]_{\text{fuz}}) = \emptyset$). We have $k_1 = 2, k_2 = 3, k_4 = 6$ for the first hit and $k_1 = 6, k_2 = 7, k_3 = 10$ for the second hit. In the case of the first hit, for example, this means that V_1 corresponds to segment 2 of $F[D]$, V_2 to segments 3–5, and V_3 to segment 6, amounting to a coarsened segmentation of D.

The fuzzy query in the previous example was chosen in such a way that the adaptive fuzzy hits with respect to $F^2 = (F^r, F^l)$ correspond to the exact hits with respect to the feature function F^r, which is the first component of F^2. More generally, it is easily verified that restricting the feature function $F : \mathscr{P} \to \{0,1\}^f$ to some of its f components can be simulated by a special choice of fuzzy sets in adaptive fuzzy searching.

Fuzzy search can be complemented by the concept of *m-mismatch search*, introducing another degree of inexactness. An m-mismatch hit permits up to $m < |F[Q]|$ feature segments within $F[Q]$ to disagree with the database, i.e., it permits up to m of the demanded equalities in (4) to be unsatisfied. To prevent arbitrary matches, it is possible to restrict the positions within $F[Q]$ where such mismatches may occur. m-mismatch hits allow for spatial variations via deviating feature values, whereas the temporal structure of the query must be matched exactly. Efficient dynamic programming algorithms for m-mismatch queries exist, see Clausen and Kurth [2004] for further details.

5.3 Indexing

Recall that the index $I^{\mathscr{D}}_F$ as introduced in Sect. 5.1 consists of 2^f inverted lists $L(v)$, each of which corresponds to the feature vector $v \in \{0,1\}^f$. The lists contain segment positions of the F-segmentation (rather than frame positions), and each segment position appears in exactly one list. As a consequence, the index size is roughly proportional to the number of segments of all documents in the database $\mathscr{D}$ (in practice, we also store the segment lengths so the frame positions can be recovered). In view of efficiency, we have to deal with the problem that the number 2^f of inverted lists is far too large in practice. For example, for our case of $f = 31$ we obtain 2^{31} lists. Typically, the segment positions distribute over a large number of different lists, which makes the retrieval process computationally expensive due to the large number of required merging and intersecting operations. To cope with this problem, we proceed as follows: we divide the set of our 31 boolean features into the three sets F_ℓ (11 features), F_{u} (12 features), and F_{m} (8 features) as indicated by Table 1. Identifying the sets with the corresponding feature function, we then construct separate indexes $I^{\mathscr{D}}_\ell$, $I^{\mathscr{D}}_{\text{u}}$, and $I^{\mathscr{D}}_{\text{m}}$. Then, retrieval amounts to querying the individual

Index	f	2^f	#(lists)	#(frames)	#(segs)	MB	$\frac{\text{bytes}}{\text{seg}}$	t_r	t_f	t_i	$\sum t$
I_ℓ^{60}	11	2048	409	425,294	21,108	0.72	35.8	26	10	6	42
I_ℓ^{180}	11	2048	550	1,288,846	41,587	1.41	35.5	71	26	13	110
I_u^{60}	12	4096	642	425,294	53,036	1.71	33.8	26	13	10	49
I_u^{180}	12	4096	877	1,288,846	135,742	4.33	33.4	71	33	25	129
I_m^{60}	8	256	55	425,294	19,067	0.60	33.0	26	20	3	49
I_m^{180}	8	256	75	1,288,846	55,526	1.80	34.0	71	54	12	137

Table 2: Feature computation and index construction. Running times are in seconds.

Type, #(segs)	1–9 hits			10–99 hits			$\geq$ 100 hits		
	μ_h	σ_h	μ_t (ms)	μ_h	σ_h	μ_t (ms)	μ_h	σ_h	μ_t (ms)
exact, $\|Q\| = 5$	3.0	2.4	16	44	28	20	649	567	144
exact, $\|Q\| = 10$	1.7	1.6	17	34	22	26	239	147	71
exact, $\|Q\| = 20$	1.1	0.6	19	32	26	36	130	5	52
fuzzy, $\|Q\| = 5$	3.6	2.5	23	44	27	29	1,878	1,101	291
fuzzy, $\|Q\| = 10$	2.4	2.1	28	40	26	35	1,814	1,149	281
fuzzy, $\|Q\| = 20$	2.0	1.9	42	36	24	35	1,908	1,152	294

Table 3: Statistics on 10,000 random queries in I_u^{180} using different query modes and query sizes, grouped by the number of hits, h. μ_h and σ_h are the average/standard deviation of h for the respective group, μ_t is the average query time in milliseconds.

indexes and post-processing the resulting hits by additional merging and/or intersecting operations. However, the number of such additional operations is by far outweighed by the savings resulting from the significantly reduced overall number $(2^{11}+2^{12}+2^8)$ of inverted lists. Furthermore, list lengths are in practice well-balanced and medium-sized, allowing for fast merging and intersecting operations. A minor drawback is that the indexes $I_\ell^{\mathscr{D}}$, $I_\mathrm{u}^{\mathscr{D}}$, and $I_\mathrm{m}^{\mathscr{D}}$ require an amount of memory linear in the overall number of F_ℓ-, F_u-, and F_m-segments in $\mathscr{D}$, respectively. This effect, however, is attenuated by the fact that segment lengths with respect to F_ℓ-, F_u-, and F_m are generally larger compared to F-segment lengths, resulting in fewer segments.

6 Experiments and Results

6.1 Indexing

We implemented our indexing and retrieval algorithms in Matlab and tested them on a database $\mathscr{D}^{180}$ containing more than one million frames of motion capture data (180 minutes sampled at 120 Hz). The experiments were run on a 3.6 GHz Pentium 4 with 1 GB of main memory. The resulting indexes are denoted by I_ℓ^{180}, I_u^{180}, and I_m^{180}. In its columns, Table 2 shows the number f of feature components, the number 2^f of inverted lists, the number of nonempty inverted lists, the overall number of frames in the database, the overall number of segments of the corresponding segmentation, the index size in MB, the number of bytes per segment, and four running times t_r, t_f, t_i, and $\sum t$, measured in seconds. t_r is the portion of running time spent on data read-in, t_f is the feature extraction time, t_i is the inverted list build-up time, and $\sum t$ is the total running time. To demonstrate the scalability of our result, we quote analogous numbers for the indexes I_ℓ^{60}, I_u^{60} and I_m^{60} built from a subset $\mathscr{D}^{60}$ of $\mathscr{D}^{180}$ corresponding to 60 minutes of motion capture data. The total size of $\mathscr{D}^{180}$ represented in the text-based AMC motion capture file format was 600 MB, a more compact binary double precision representation required about 370 MB. Typical index sizes ranged between 0.7 and 4.3 MB, documenting the drastic amount of data reduction our scheme achieves.

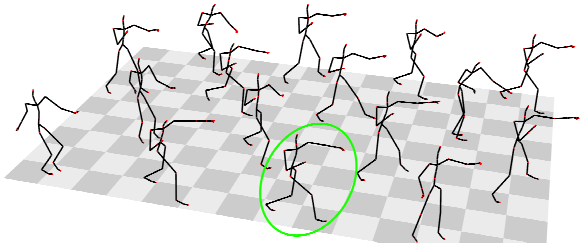

Figure 7: Selected frames from 16 query-by-example hits for a left hand punch. The query clip is highlighted. Query features: F_u^1, F_u^2, F_u^7, F_u^8; see Table 1.

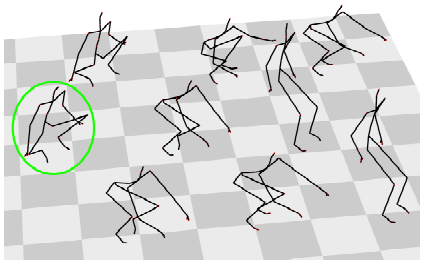

Figure 8: Selected frames from 9 query-by-example hits for a squatting motion. The query clip is highlighted. Query features: F_ℓ^5, F_ℓ^6, F_ℓ^7, F_ℓ^{10}, F_ℓ^{11}; see Table 1.

Table 2 shows that the number of segments (with respect to F_ℓ, F_u, and F_m) was only about 3 to 12 percent of the number of frames contained in the database. Observe that index sizes are proportional to the number of segments: the average number of bytes per segment is constant for all indexes. The total indexing time is *linear* in the number of frames. This fact is very well reflected in the table: for example, it took 42 seconds to build I_ℓ^{60}, which is roughly one third of the 110 seconds that were needed to build I_ℓ^{180}. Note that more than half of the total indexing time was spent on reading in the data, e.g., 71 seconds for the 180-minute index. The scalability of our algorithms' running time and memory requirements permits us to use much larger databases than those treated by Kovar and Gleicher [2004], where the preprocessing step to build a match web is quadratic in the number of frames (leading, e.g., to a running time of roughly 3,000 seconds for a database containing only 37,000 frames).

6.2 Retrieval

The running time to process a query very much depends on the size of the database, the query length (the number of segments), the user-specified fuzziness of the query, as well as the number of resulting hits. In an experiment, we posed 10,000 random queries (guaranteed to yield at least one hit) for each of six query scenarios to the index I_u^{180}, see Table 3. For example, finding all exact F_u-hits for a query consisting of $5/10/20$ segments takes on average 16–144/17–71/19–52 milliseconds, depending on the number of hits. Finding all adaptive fuzzy F_u-hits for a query consisting of $5/10/20$ segments, where each fuzzy set of alternatives has a size of 64 elements, takes on average 23–291/28–281/42–294 ms.

Fig. 7 depicts all resulting 16 hits from a query for a punch (retrieval time: 12.5 ms), where only the four features $F_\mathrm{u}^1/F_\mathrm{u}^2$ (right/left hand in front) and $F_\mathrm{u}^7/F_\mathrm{u}^8$ (right/left elbow bent) have been selected, see Table 1. These four features induce an adaptive segmentation of the query consisting of six segments, which suffice to grasp the gist of the punching motion. Further reducing the number of features by

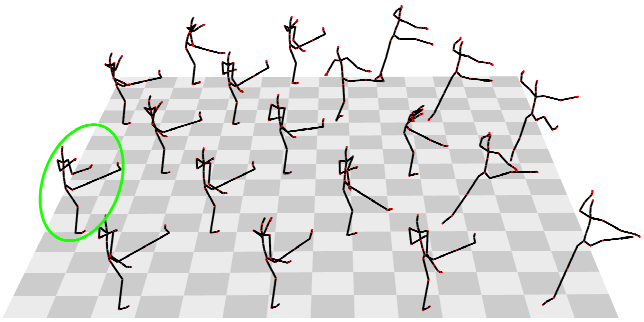

Figure 9: Selected frames from 19 query-by-example hits for a right foot kick. The query clip is highlighted. Query features: $F_\ell^3, F_\ell^4, F_\ell^7, F_\ell^8$; see Table 1.

selecting only F_u^2 and F_u^8 induces a 4-segment query sequence and results in 264 hits, comprising various kinds of punch-like motions involving both arms. Finally, increasing the number of selected features by adding F_u^3/F_u^4 induces an 8-segment query sequence resulting in a single hit.

Fig. 8 shows 9 hits out of the resulting 33 hits for a "squatting" motion (retrieval time: 18 ms) using the five features $F_\ell^5, F_\ell^6, F_\ell^7, F_\ell^{10}$, and F_ℓ^{11}. The induced 5-segment query sequence is characteristic enough to retrieve 7 of the 11 "real" squatting motions contained in the database. Using a simple ranking strategy (e.g., a temporal heuristic comparing the sum of frame length differences between query and hit segments), these 7 hits appear as the top hits. The remaining 26 retrieved hits are false positives, two of which are shown to the right of Fig. 8 as the skeletons "sitting down" on a virtual table edge. One reason for this kind of false positives is that the relevant feature used in the query for the squatting motion thresholds the knee angle against a relatively high decision value of 120°. Hence, the knees of the sitting skeletons are just barely classified as "bent," leading to the confusion with a squatting motion. Omitting the velocity features F_ℓ^5 and F_ℓ^6 again results in an induced 5-segment query, this time, however, yielding 63 hits (containing the previous 33 hits with the same top 7 hits). Among the additional hits, one now also finds jumping and sneaking motions.

Finally, Fig. 9 shows all 19 query results for a "kicking" motion (retrieval time: 5 ms) using $F_\ell^3, F_\ell^4, F_\ell^7$, and F_ℓ^8. Out of these, 13 hits are actual martial arts kicks. The remaining six motions are ballet moves containing a kicking component. A manual inspection of $\mathscr{D}^{180}$ showed that there are no more than the 13 reported kicks in the database, demonstrating the high recall percentage our technique can achieve. Again, reducing the number of selected features leads to an increased number of hits. In general, a typical source of false positive hits is the choice of fuzzy alternatives in a query. For example, the ballet jumps in Fig. 9 were found as matches for a kicking motion because only the right leg was constrained by the query, leaving the left leg free to be stretched behind the body.

In conclusion, our technique can efficiently retrieve high-quality hits with good precision/recall percentages provided that the user adequately selects a small number of features reflecting the important aspects of the query motion. This, in general, requires some experience as well as parameter tuning. However, most features have strong semantics, which makes feature selection a very intuitive process, see Sect. 6.3. To automate feature selection, we propose a hierarchical approach: starting with a family of predefined feature sets, systematically reduce the number of features in these sets and pose a query with respect to each resulting feature combination. This process is repeated until a sufficient number of hits has been retrieved. Since this approach will generally produce a large percentage of false positives—often exceeding 80%—one needs suitable ranking strategies. Here, one could refine the retrieval results by applying more involved DTW-based techniques, see Sect. 7.

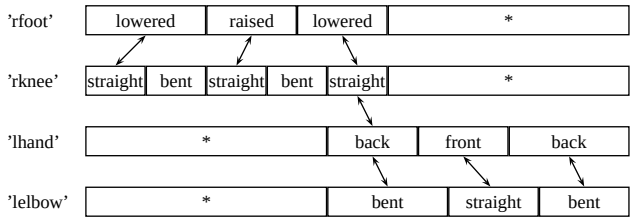

Figure 10: Scene description for the movement "right foot kick followed by a left hand punch" as shown in Fig. 4. The symbol * indicates that a constellation is unspecified. Arrows indicate which constellations are to occur simultaneously.

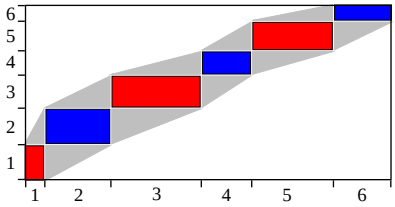

Figure 11: The segment-wise time alignment of the two walking motions shown in Fig. 5 (corresponding to the vertical axis) and Fig 6 (corresponding to the horizontal axis) can be refined via DTW-based methods restricted to the gray area.

6.3 Towards Scene Descriptions

Often, the user will only have a sketchy idea of which kind of movement to look for in the motion capture database, for example a "right foot kick followed by a left hand punch." That is, the query may not be given as an example motion but only be specified by a vague textual description. In processing such queries, one so far has to revert to annotations such as manually generated descriptive labels attached to the motion data. Our indexing strategy, on the other hand, is purely content-based and thus facilitates fully automated preprocessing. Furthermore, it provides an efficient way of focusing on different aspects of the motion and selecting the desired search granularity *after* the preprocessing stage (and not *before* as for manual annotations). First experiments to implement sketchy queries such as the one above with our retrieval technique are based on the following observation: vaguely specified motions such as kicking, punching, or clapping can often be specified by a very small set of basic geometric constellations. For example, as depicted in Fig. 10, a kick of the right foot can be quite well characterized by the concurrent progression of the constellations "rknee straight/bent/straight/bent/straight" and "rfoot lowered/raised/lowered." In addition, the intermediary constellations "rknee bent" and "rfoot raised" should overlap at some point in time. Similarly, Fig. 10 shows a *geometric scene description* for the motion "right foot kick followed by a left hand punch," where overlapping constellations are indicated by an arrow and unspecified constellations, realizable by our fuzzy concept, are indicated by the symbol *. Now, such kinds of geometric scene descriptions can be efficiently processed by first querying for each progression separately and then suitably intersecting the retrieved hits to account for cooccurrence conditions. In our future work, we plan to employ statistical methods to learn such geometric scene descriptions from example motions to fully automate queries at higher semantic levels, see also Sect. 8.

7 Time Alignment

Recall that two motion clips are considered as similar if they possess (more or less) the same progression of geometric features. Matching two such progressions obtained from similar motions can be regarded as a time alignment of the underlying motion data streams. Even though such alignments may be too coarse in view of applications such as morphing or blending, they are quite accurate with respect to the overall course of motion. For the time alignment of motion data streams we therefore suggest the following two-stage procedure: first compute (in linear time) a coarse segment-wise alignment based on our index. Then refine this alignment resorting to classical DTW techniques, as, e.g., described by Kovar and Gleicher [2004]. The important point is that once a coarse alignment is known, the DTW step can be done very efficiently since the underlying cost matrix need only be computed within an area corresponding to the frames of the matched segments. This is also illustrated by Fig. 11, where the two walking motions of Fig. 5 and Fig 6 are aligned. In order to avoid alignment artifacts enforced by segment boundaries, the restricted area is slightly enlarged as indicated by the gray area. For more details on accelerating DTW computations we refer to, e.g., Keogh [2002].

8 Conclusions and Future Work

This paper has presented automated methods for efficient indexing and content-based retrieval of motion capture data. One main contribution of this work is the introduction of qualitative, geometric features—opposed to quantitative, numerical features used in previous approaches. A second contribution is the concept of adaptive temporal segmentation, by which segment lengths are not only adjusted to the granularity of the feature function but also to the fuzziness of the query. It is the combination of geometric features and induced segmentations that accounts for spatio-temporal invariance, which is crucial for the identification of logically related motions. Thirdly, we have adapted the notion of fault-tolerant retrieval based on fuzzy hits and mismatches that can be efficiently computed by means of inverted lists. One decisive advantage of our index structure is that the time as well as the space to construct and store the index is linear in the size of the database. This solves the problem of scalability emerging in DTW-based approaches. We have also sketched how our methods can be applied in a preprocessing step to accelerate DTW-based time alignment of motion capture data streams.

Future Work: One major drawback in our query scenario is that for each query the user has to select suitable features in order to obtain high-quality retrieval results. This is not feasible in case one wants to batch process many different motion clips, as necessary in morphing and blending applications. However, as Sect. 7 showed, our technique should not be seen as a mere alternative but rather as a complement to previous DTW-based techniques. Using simple geometric relations avoiding false dismissals often helps to cut down the search space significantly without loss of quality so that more involved cost-intensive methods may be applied for post-processing the restricted data set. Furthermore, as first experiments showed, geometric features also seem to be a promising tool in handling low-level descriptive queries automatically. Here, the idea is that movements can often be characterized by a typical progression of geometric constellations corresponding to key poses. We plan to employ statistical methods to learn such progressions as well as to automatically identify expressive geometric features from typical example motions. Conversely, such progressions can then aid in automatically annotating newly generated motion capture data. In view of such applications, one first has to build a high-quality manually annotated database comprising various kinds of example motions. It would be extremely valuable to the research community if publicly available motion databases such as CMU [2003] were extended in that way, making experimental results comparable. For example, lacking such a common database, it was not possible for us to objectively compare and combine our retrieval results with those of related techniques such as that of Kovar and Gleicher [2004].

Acknowledgements: We would like to thank Andreas Weber for his encouragement and for helpful discussions, as well as the anonymous referees for constructive and valuable comments. Some of the data used in this project was obtained from `mocap.cs.cmu.edu`, which was created with funding from NSF EIA-0196217. Some of the data was obtained from Modern Uprising Studios.

References

BAKKER, E. M., HUANG, T. S., LEW, M. S., SEBE, N., AND ZHOU, X. S., Eds. 2003. Proc. 2nd Intl. Conf. Image and Video Retrieval, CIVR 2003, Urbana-Champaign, IL, USA, vol. 2728 of *LNCS*, Springer.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proc. ACM SIGGRAPH 95*, ACM Press / ACM SIGGRAPH, Computer Graphics Proc., Annual Conf. Series, 97–104.

CARDLE, M., VLACHOS, S. B., KEOGH, E., AND GUNOPULOS, D. 2003. Fast motion capture matching with replicated motion editing. ACM SIGGRAPH 2003, Sketches and Applications.

CARLSSON, S. 1996. Combinatorial geometry for shape representation and indexing. In *Object Representation in Computer Vision*, 53–78.

CARLSSON, S. 1999. Order structure, correspondence, and shape based categories. In *Shape, Contour and Grouping in Computer Vision*, Springer, 58–71.

CLAUSEN, M., AND KURTH, F. 2004. A unified approach to content-based and fault tolerant music recognition. *IEEE Transactions on Multimedia 6*, 5, 717–731.

CMU, 2003. Carnegie-Mellon MoCap Database. `http://mocap.cs.cmu.edu`.

GIESE, M., AND POGGIO, T. 2000. Morphable models for the analysis and synthesis of complex motion patterns. *IJCV 38*, 1, 59–73.

GREEN, R., AND GUAN, L. 2004. Quantifying and recognizing human movement patterns from monocular video images: Part I. *IEEE Transactions on Circuits and Systems for Video Technology 14*, 2 (February), 179–190.

KEOGH, E. J., PALPANAS, T., ZORDAN, V. B., GUNOPULOS, D., AND CARDLE, M. 2004. Indexing large human-motion databases. In *Proc. 30th VLDB Conf., Toronto*, 780–791.

KEOGH, E. 2002. Exact indexing of dynamic time warping. In *Proc. 28th VLDB Conf., Hong Kong*, 406–417.

KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curves. In *Proc. ACM SIGGRAPH 2003 / Eurographics Symposium on Computer Animation*, Eurographics Association, 214–224.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3, 559–568.

LAST, M., KANDEL, A., AND BUNKE, H., Eds. 2004. *Data Mining In Time Series Databases*. World Scientific.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics*, 501–508.

SULLIVAN, J., AND CARLSSON, S. 2002. Recognizing and tracking human action. In *ECCV '02: Proc. 7th European Conf. on Computer Vision—Part I*, 629–644.

WITKIN, A., AND POPOVIC, Z. 1995. Motion warping. In *Proc. ACM SIGGRAPH 95*, ACM Press / ACM SIGGRAPH, Computer Graphics Proc., Annual Conf. Series, 105–108.

WITTEN, I. H., MOFFAT, A., AND BELL, T. C. 1999. *Managing Gigabytes*. Morgan Kaufmann Publishers.

WU, M.-Y., CHAO, S., YANG, S., AND LIN, H. 2003. Content-based retrieval for human motion data. In *16th IPPR Conf. on Computer Vision, Graphics and Image Processing*, 605–612.