Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2009)
E. Grinspun and J. Hodgins (Editors)

Efficient and Robust Annotation of Motion Capture Data

Meinard Müller, Andreas Baak, Hans-Peter Seidel

Saarland University and MPI Informatik
Campus E1 4, 66123 Saarbrücken, Germany
{meinard,abaak,hpseidel}@mpi-inf.mpg.de

Abstract

In view of increasing collections of available 3D motion capture (mocap) data, the task of automatically annotating large sets of unstructured motion data is gaining in importance. In this paper, we present an efficient approach to label mocap data according to a given set of motion categories or classes, each specified by a suitable set of positive example motions. For each class, we derive a motion template that captures the consistent and variable aspects of a motion class in an explicit matrix representation. We then present a novel annotation procedure, where the unknown motion data is segmented and annotated by locally comparing it with the available motion templates. This procedure is supported by an efficient keyframe-based preprocessing step, which also significantly improves the annotation quality by eliminating false positive matches. As a further contribution, we introduce a genetic learning algorithm to automatically learn the necessary keyframes from the given example motions. For evaluation, we report on various experiments conducted on two freely available sets of motion capture data (CMU and HDM05).

Categories and Subject Descriptors (according to ACM CCS):
Information Storage and Retrieval [H.3.3]: Information Search and Retrieval—Three-Dimensional Graphics and Realism [I.3.7]: Animation—

1. Introduction

The usage of prerecorded human 3D motion capture (mocap) data to create naturally looking motion sequences has become a standard procedure in computer animation [PB02, AFO03, KG04, CH05, CHP07]. Current data-driven motion controllers allow for generating a wide range of task-specific motion sequences satisfying additional spatial and temporal constraints. Most of the proposed controllers are built upon carefully compiled sets of prototype motions that cover the desired range of tasks and execution modes. Acquisition and capturing of suitable motions for building up such specialized data sets is a labor and cost intensive task [CHP07]. Therefore, various strategies have been described to reuse previously recorded motions stored in a database. In this context, a thorough and reliable annotation of the stored motions is of great importance. Even though there is a rapidly growing corpus of freely available mocap data [CMU03, MRC*07], there is still a lack of efficient systems that automatize the annotation process without manual intervention. Here, one main challenge is to deal with significant spatial as well as temporal variations that may be present in semantically related motions [KG04, Mül07]. For a discussion of related work, we refer to Sect. 2.

In this paper, we present a system for automatically and efficiently annotating large unstructured collections of mocap data. Given an unknown mocap document, the annotation task consists of segmenting the document into logical units and then locally classifying each segment according to a given set of motion classes. Here, note that the problem of *locally* annotating unknown motion data on the subsegment level is a much harder task than *globally* comparing and classifying motion data on the document level. In our annotation scenario, we assume that each motion class is specified by a set of semantically related example motions which reflect the range of spatio-temporal variations appearing in valid motion realizations. As motion class representation, we revert to the concept of *motion templates* (MTs) as introduced by Müller et al. [MR06]. Such templates capture common as well as varying aspects of the underlying training motions in

an explicit and semantically interpretable matrix representation.

As a first main contribution of this paper, we describe a novel MT-based annotation procedure to segment and label an unknown motion document on the basis of a given set of motion classes. Here, an assigned label corresponds to the motion class that best explains the respective motion segment. Unlike previous work, our annotation procedure shows a high degree of robustness to large numerical differences that may exists between semantically related motions (i. e., motions that belong to the same motion class). Furthermore, we show how the annotation procedure can be assisted by a keyframe-based search algorithm, which not only efficiently narrows the set of candidate motions related to a specific motion class but also improves the annotation quality by eliminating false positive matches. Intuitively, we first prune the unknown motion document using a fast keyframe-based search. Hereafter, the MT-based annotation procedure is conducted only on a small subset of the document.

As another major contribution of this paper, we describe a genetic algorithm that allows for learning characteristic keyframes in a fully automatic evolutionary process using positive and negative example motions. Finally, to demonstrate the practicability of our overall annotation procedure, we describe various experiments conducted on motion documents obtained from the two freely available motion capture databases HDM05 [MRC*07] and CMU [CMU03].

The remainder of this paper is organized as follows. We discuss related work (Sect. 2) and review the concept of motion templates (Sect. 3). Then, in Sect. 4, we describe our novel MT-based annotation procedure. In Sect. 5, we show how efficiency and precision can be significantly improved by employing a keyframe-based preselection step. Furthermore, in Sect. 6, we introduce a genetic algorithm for deriving the necessary keyframes from positive and negative example motions. Finally, in Sect. 7, we report on our experiments and conclude in Sect. 8 with prospects on future work.

2. Related Work

In the last years, various retrieval and classification algorithms have been proposed to automate the annotation process, see, e. g., [WCYL03, KPZ*04, KG04, LZWM05, FF05, MRC05, MR06]. Here, the main difficulty arises from the fact that semantically similar motions may reveal significant numerical differences [KG04, Mül07]. Most of the above cited procedures use motion representations that are semantically close to the raw data. Here, problems occur when one has to cope with strong pose deformations within a class of logically related motions. Approaches such as [LZWM05, MRC05] absorb spatial and temporal variations already on the feature level, which then allows for a more robust and efficient motion comparison. In our annotation procedure, we use the concept of motion templates as introduced by Müller et al. [MR06], which allows for grasping the essence of an entire class of motions within an explicit matrix representation. Several approaches to classification and recognition of motion patterns are based on HMMs, which are also a flexible tool to capture spatio-temporal variations, see, e. g., [BH00]. Temporal segmentation of motion data can be viewed as another form of annotation, where consecutive, logically related frames are organized into groups, see, e. g., [BSP*04].

The use of prerecorded motion capture data to create new realistic motions has become a standard technique in data-driven computer animation, see, e. g., [PB02, AFO03, KG04, CH05, CHP07]. In view of motion synthesis applications, one needs specialized and controlled data sets which are often obtained from manually annotated material. For example, Rose et al. [RCB98] group similar example motions into "verb" classes to synthesize new, user-controlled motions by suitable interpolation techniques. For synthesizing new motions from motion graphs, Kovar et al. [KGP02] allow the use of annotation constraints. Arikan et al. [AFO03] propose a semi-automatic annotation procedure, where a user is required to annotate only a small portion of the database. The user annotations are then generalized to the entire database in a framewise fashion using SVM classifiers. Our annotation approach differs from their approach in various ways. First, our annotation strategy is segment-based instead of frame-based, thus resulting in semantically more meaningful units. Second, using concepts such as relational features and dynamic time warping, our approach is more robust to spatial and temporal variations than the one by Arikan et al. [AFO03], where normalized joint positions and fixed temporal windows are used. Finally, our strategy is to learn the necessary class representations (motion templates, keyframes) only once prior to the actual annotation step. Based on these representations, the annotation can then be performed very efficiently on large and arbitrary sets of mocap documents.

3. Motion Templates

We now review the concept of motion templates, as introduced by Müller et al. [MR06]. As underlying feature representation, *relational features* are employed to capture semantically meaningful boolean relations between specified points of the kinematic chain underlying the mocap data [MRC05]. The main point is that even though relational features discard a lot of detail contained in the raw motion data, important information regarding the overall configuration of a pose is retained. In the following, we use a set of $f = 40$ relational features, where the first 39 features are defined as in [MR06] and the last feature expresses whether the angular velocity of the root orientation is high or not.

Now, given a class $\mathcal{C}$ consisting of $\gamma \in \mathbb{N}$ example motions, such as the four motions from the class 'sitDownFloor' shown in Fig. 1 (a), the goal is to automatically learn a mo-

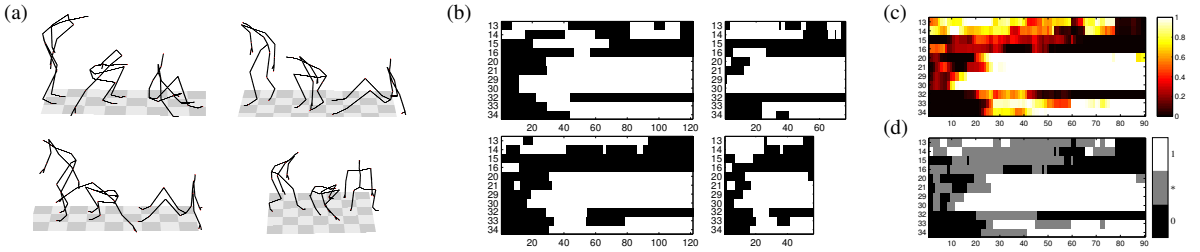

Figure 1: ***(a)*** *Selected frames from four different motions of the class 'sitDownFloor'.* ***(b)*** *Resulting boolean feature matrices for selected relational features (numbered in accordance with the features defined in [MR06]). The columns represent time in frames (using 30 fps), whereas the rows correspond to boolean features encoded as black (0) and white (1).* ***(c)*** *Class MT for 'sitDownFloor' based on the $\gamma = 4$ training motions shown in (a).* ***(d)*** *Corresponding quantized class MT.*

tion class representation that grasps the essence of the class. One starts by computing the relational feature vectors for each of the γ motions. Denoting the length of a given motion by K, the resulting sequence of feature vectors can be thought of as a *feature matrix* $X \in \{0,1\}^{f \times K}$ as shown in Fig. 1 (b), where, for the sake of clarity, we display a subset comprising only eleven of the $f = 40$ features.

Next, a semantically meaningful average over the γ feature matrices is computed. To cope with temporal variations in the example motions, an iterative warping and averaging algorithm is employed, which converges to an output matrix $X_{\mathcal{C}}$ referred to as a *motion template* (MT) for the class $\mathcal{C}$. The matrix $X_{\mathcal{C}}$ has real-valued entries between zero and one and has a length (number of columns) corresponding to the average length of the training motions. Fig. 1 (c) shows a motion template obtained from $\gamma = 4$ motions of the class 'sitDownFloor'. The important observation is that black/white regions in a class MT indicate periods in time (horizontal axis) where certain features (vertical axis) consistently assume the same values zero/one in all training motions, respectively. By contrast, colored regions indicate inconsistencies mainly resulting from variations in the training motions. In other words, the black/white regions encode characteristic aspects that are shared by all motions, whereas the colored regions represent the class variations coming from different realizations. Finally, one obtains a *quantized MT* by replacing each entry of $X_{\mathcal{C}}$ that is below a quantization threshold δ by zero, each entry that is above $1-\delta$ by one, and all remaining entries by a *wildcard character* $*$ indicating that the corresponding value is left unspecified, see Fig. 1 (d). In our annotation experiments (Sect. 7), we use the threshold $\delta = 0.05$, which has turned out to yield a good trade-off between robustness to motion variations and discriminative power. Only in the keyframe-based learning procedure, where quantized MTs are employed to initialize a learning procedure to derive hard keyframe constraints, we use the strict quantization threshold $\delta = 0$, see Sect. 6.

4. Annotation Procedure

As basis for our annotation procedure, we introduce a distance function that reveals all motion subsegments of an unknown mocap document D associated with a given motion class $\mathcal{C}$. Let $X \in \{0,1,*\}^{f \times K}$ be the quantized class MT of $\mathcal{C}$ of length K and $Y \in \{0,1\}^{f \times L}$ the feature matrix of D of length L. We first define a cost measure c^Q, which allows for comparing the k^{th} column $X(k)$ of X and the ℓ^{th} column $Y(\ell)$ of Y, $k \in [1:K]$, $\ell \in [1:L]$. Let $I(k) := \{i \in [1:f] \mid X(k)_i \neq *\}$, where $X(k)_i$ denotes the i^{th} entry of the k^{th} column of X. Then, if $|I(k)| > 0$, we set

$$c^Q(k,\ell) = \frac{1}{|I(k)|} \sum_{i \in I(k)} |X(k)_i - Y(\ell)_i|, \tag{1}$$

otherwise we set $c^Q(k,\ell) = 0$. In other words, $c^Q(k,\ell)$ only accounts for the consistent entries of X with $X(k)_i \in \{0,1\}$ and leaves the other entries unconsidered. Based on this cost measure, we define a distance function $\Delta : [1:L] \to \mathbb{R} \cup \{\infty\}$ between X and Y using dynamic time warping (DTW):

$$\Delta(\ell) := \frac{1}{K} \min_{a \in [1:\ell]} \Big(\mathrm{DTW}\big(X, Y(a:\ell)\big) \Big), \tag{2}$$

where $Y(a:\ell)$ denotes the subsequence of Y starting at index a and ending at index $\ell \in [1:L]$. Furthermore, $\mathrm{DTW}(X, Y(a:\ell))$ denotes the DTW distance between X and $Y(a:\ell)$ with respect to the cost measure c^Q. To avoid degenerations in the DTW alignment, we use the modified step size condition with step sizes $(2,1)$, $(1,2)$, and $(1,1)$ (instead of the classical step sizes $(1,0)$, $(0,1)$, and $(1,1)$). Note that the distance function Δ can be computed efficiently using dynamic programming. For details on DTW and the distance function, we refer to [Mül07]. The interpretation of Δ is as follows: a small value $\Delta(\ell)$ for some $\ell \in [1:L]$ indicates that the subsequence of Y starting at frame a_ℓ (with $a_\ell \in [1:\ell]$ denoting the minimizing index in Eq. (2)) and ending at frame ℓ is similar to the class MT X. In other words, looking for all local minima in Δ below a suitable quality threshold $\tau > 0$ one can identify all subsegments of D closely correlating to the class $\mathcal{C}$. As example, Fig. 2 (a) shows a distance function based

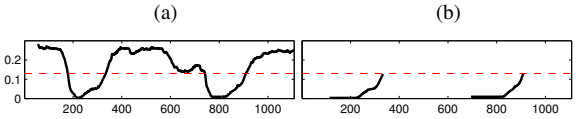

Figure 2: ***(a)*** *Distance functions* Δ *based on the quantized class MT 'sitDownFloor'* ***(b)*** *Corresponding modified distance function* $\bar{\Delta}^{\tau}$ *for* $\tau = 0.13$.

on the quantized MT for the class 'sitDownFloor'. Note that there are two local minima having a value close to zero that reveal the two 'sitDownFloor' subsegments contained in the mocap document.

Recall that a local minimum $\Delta(\ell)$ close to zero only indicates the *end frame* of a subsegment of D corresponding to the class $\mathcal{C}$. We now modify the distance function in such a way that *all frames* $k \in [a_\ell : \ell]$ of the subsegment are distinguished by assigning to them the same distance value $\Delta(\ell)$. Furthermore, only those frames should be considered that closely correlate to $\mathcal{C}$. To this end, we introduce a quality threshold $\tau > 0$ and iteratively define a modified distance function $\bar{\Delta}^{\tau} : [1:L] \to \mathbb{R} \cup \{\infty\}$. First, we set $\bar{\Delta}^{\tau}(k) = \infty$ for all $k \in [1:L]$. Then, iterating over all local minima $\ell \in [1:L]$ of Δ below τ, we define $\bar{\Delta}^{\tau}(k)$ for $k \in [a_\ell : \ell]$ to be the minimum of the hitherto defined value $\bar{\Delta}^{\tau}(k)$ and $\Delta(\ell)$, see Fig. 2 (b).

The basic idea of our annotation procedure is to locally compare a mocap document with the various class motion templates and then to annotate all frames within a suitable motion segment with the label of the motion class that best explains the segment. Let D be an unkown mocap document of length L and let $\mathcal{C}_1, \ldots \mathcal{C}_P$ be the motion classes used for the annotation, where $p \in [1:P]$ denotes the label of class $\mathcal{C}_p$. In our procedure, we compute a modified distance function $\bar{\Delta}^{\tau}_p$ for each class $\mathcal{C}_p$ as described above. We then minimize the resulting functions over all class labels $p \in [1:P]$ to obtain a single function $\Delta^{\min} : [1:L] \to \mathbb{R} \cup \{\infty\}$:

$$\Delta^{\min}(\ell) := \min_{p \in [1:P]} \bar{\Delta}^{\tau}_p(\ell), \tag{3}$$

$\ell \in [1:L]$. Furthermore, we store for each frame the minimizing index $p \in [1:P]$ yielding a function $\Delta^{\arg} : [1:L] \to [0:P]$ defined by:

$$\Delta^{\arg}(\ell) := \operatorname*{argmin}_{p \in [1:P]} \bar{\Delta}^{\tau}_p(\ell), \tag{4}$$

where $\Delta^{\arg}(\ell)$ is set to 0 in case $\Delta^{\min}(\ell) = \infty$ (and to the minimal class label number to break a tie). In principle, the function $\Delta^{\arg}$ yields the annotation of the mocap document D by means of the class labels $p \in [1:P]$. Here, a value 0 means that the corresponding frame is left unannotated.

For a first illustrative example, we use the $P = 15$ classes indicated by Table 1. Fig. 3 (a) shows the resulting 15 modified distance functions $\bar{\Delta}^{\tau}_p$ with $\tau = 0.13$ in a color-coded

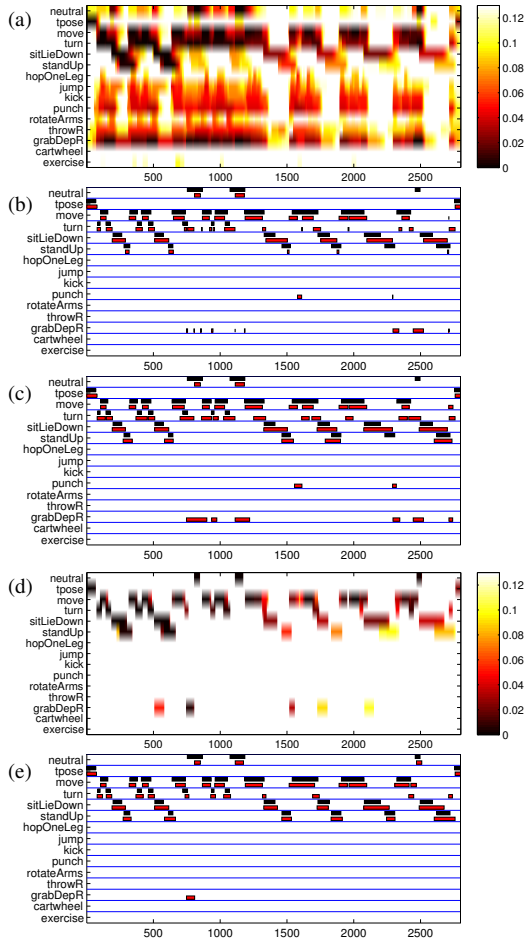

Figure 3: ***(a)*** *Modified distance functions (color coded) for each class. White regions indicate distance values larger than* $\tau = 0.13$ ***(b)*** *Red blocks: Resulting annotations induced by* $\Delta^{\arg}$*. Black blocks: Ground truth annotations.* ***(c)*** *Result after extending (b).* ***(d)*** *Modified distance functions using keyframes as preprocessing step.* ***(e)*** *Annotation result using keyframes.*

form for a given mocap document D of length $L = 2800$ frames (≈ 93 seconds). The resulting annotations are shown in Fig. 3 (b), where the color red corresponds to the automatically generated annotations induced by $\Delta^{\arg}$ and the color black corresponds to manually generated ground-truth annotations. See Sect. 7 for a further discussion and evaluation of our annotation results.

In the following, a maximal run of consecutive frames annotated by the same label is referred to as segment. Note that our procedure cuts the document D into disjoint segments, where some of these may be very short. For example, the 'standUp' annotation segment around frame 1500 com-

prises only 13 frames ($\approx 1/3$ sec). This is due to the fact that the beginning of the actual 'standUp'-motion (actor is sitting) is annotated as 'sitLieDown'. This makes sense since the beginning of the 'standUp' motion semantically overlaps with the end of the previous 'sitLieDown'-motion, where the actor is sitting down. To allow for overlapping annotations and semantically meaningful segments (i.e., segments that represent a complete motion of the corresponding class), we further extend the annotated segments as follows: Suppose that the frames with indices $[s:t]$, $s,t \in [1:L]$, $s \leq t$, have been annotated with the class label $p \in [1:P]$, i. e., $\forall \ell \in [s:t] : \Delta^{\arg}(\ell) = p$. Then, let $r \leq s$ be the minimal index such that $\bar{\Delta}_p$ is monotonously increasing (or constant) on the interval $[r:s]$. Similarly, let $u \geq t$ be the maximal index such that $\bar{\Delta}_p$ is monotonously decreasing (or constant) on the interval $[t:u]$. Then all frames with indices in the interval $[r:u]$ will also be annotated with p, see Fig. 3 (c).

5. Keyframe-based Preselection

As indicated by Fig. 3 (c), our annotation procedure may yield a number of false positive annotations. For example, the motion class 'grabDepR', which consists of right hand grabbing and depositing motions, causes a number of confusions with other classes. The reason is that grabbing and depositing motions are short motions and possess only few characteristic aspects—basically, the right hand is moving and nothing else happens in a consistent way. This leads to a rather unspecific class MT, which reveals small distances to many motion fragments that are actually part of other motion classes. To cope with this problem, we introduce an additional keyframe-based preprocessing step. For example, for the class 'grabDepR' one may use a few keyframes enforcing that both feet do not move while the right hand moves to the front (before grabbing) and is then pulled back (after grabbing). Using such additional keyframe constraints allows for eliminating a large number of false positive annotations and, additionally, for significantly speeding up the annotation procedure.

In the following, a *keyframe query* $(\mathbf{V},\mathbf{d})$ of length N consists of a sequence $\mathbf{V} = (V_1,\ldots,V_N)$ of keyframes and a sequence $\mathbf{d} = ((d_1^{\min},d_1^{\max}),\ldots,(d_{N-1}^{\min},d_{N-1}^{\max}))$ of distance parameters. Here, a keyframe is specified by a vector $V_n \in \{0,1,*\}^f$, $n \in [1:N]$, which describes characteristic relations of a specific pose. Such a keyframe can be thought of as a column of a quantized motion template, see Fig. 4. Furthermore, the tuple $(d_n^{\min},d_n^{\max}) \in \mathbb{N}_0 \times \mathbb{N}_0$ with $d_n^{\min} \leq d_n^{\max}$, $n \in [1:N-1]$, specifies the admissible distance (in frames) of the neighboring keyframes V_n and V_{n+1}. We say that a motion segment is *relevant* with respect to a keyframe query $(\mathbf{V},\mathbf{d})$ if it exhibits feature vectors matching the keyframe vectors given by $\mathbf{V}$ in the correct order within the distance bounds specified by $\mathbf{d}$.

Now, let us return to our annotation procedure where the objective is to annotate an unknown mocap document D with respect to given motion classes $\mathcal{C}_p$, $p \in [1:P]$. For the moment, we assume that a characteristic keyframe query $(\mathbf{V}_p,\mathbf{d}_p)$ is available for each class $\mathcal{C}_p$. (In Sect. 6, we show how such keyframe queries can be learned automatically from example motions.) In a preprocessing step, we extract all motion segments from D that are relevant with respect to $(\mathbf{V}_p,\mathbf{d}_p)$. This is done by employing the keyframe-based search algorithm as introduced by Baak et al. [BMS08], which allows for explicitly controlling the degree of admissible deformations between the queried keyframes, while being efficient using an inverted file index. Then, the distance function Δ_p is computed on the relevant segments only (setting the value to ∞ for the irrelevant frames). The resulting reduction is illustrated by comparing (d) with (a) of Fig. 3: the additional white regions in (d) correspond to irrelevant information masked out by the keyframe search. The annotations obtained from (d) are shown in Fig. 3 (e). Note that the keyframe-based preselection has several benefits. First, using additional constraints allows for eliminating many false positive annotations. Furthermore, the index-based retrieval step is ideally suited to cut down the search space to relevant subsegments, thus significantly speeding up and drastically reducing memory requirements in the subsequent steps. For details on the keyframe search algorithm we refer to [BMS08]. The effect of the keyframe-based preprocessing step on the annotation quality and performance is discussed in Sect. 7.

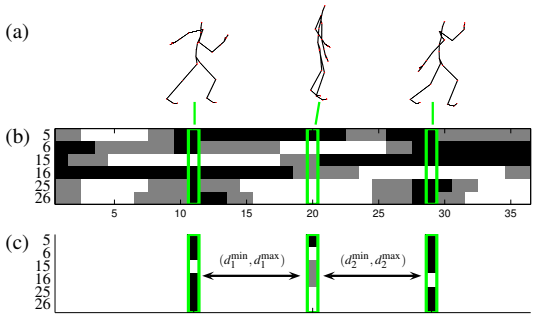

Figure 4: ***(a)****: Three characteristic key poses in a skier exercise motion.* ***(b)****: Motion template of the skier class. Columns that directly correspond to the key poses are marked green.* ***(c)****: Keyframe query.*

6. Keyframe Learning Algorithm

To improve and accelerate the overall annotation procedure, one has to carefully select and design the keyframe queries. On the one hand, since the keyframes are used as hard constraints in a query, the keyframes should generalize well to avoid a large number of false negatives in the preprocessing step. On the other hand, the keyframes must have a high discriminatory power to yield the desired pruning and data reduction capability. In this section, we describe how

characteristic keyframe queries can be learned automatically from positive and negative example motions using a randomized genetic algorithm. Generally, such algorithms are population-based optimization techniques to find approximate solutions to optimization problems [Poh99].

As in the case of motion templates, we assume a set of positive example motions $\mathcal{T}^+$ representing a motion class $\mathcal{C}$. Additionally, we assume a set of negative example motions $\mathcal{T}^-$ that discriminate the class $\mathcal{C}$ to other motion classes. Then, the goal is to generate a keyframe query $(\mathbf{V},\mathbf{d})$ yielding characteristic constraints shared by all motions belonging to $\mathcal{C}$ but not by motions from other classes. In other words, a keyframe search with $(\mathbf{V},\mathbf{d})$ conducted on the set $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$ should return exactly $\mathcal{T}^+$.

Our keyframe learning algorithm following the general paradigm of evolutionary algorithms [Poh99] is described in the following. A *population* consists of a set of *individuals* that represent candidate solutions for the optimization problem. In our scenario, an individual is a keyframe query $\mathrm{Ind} = (\mathbf{V},\mathbf{d})$. We measure the quality or the *fitness* of an individual in terms of precision and recall evaluating the individual on the example motions. More precisely, let $\mathcal{H}(\mathrm{Ind}) \subseteq \mathcal{T}$ denote the mocap documents retrieved by the keyframe query Ind. The precision $P(\mathrm{Ind})$ and recall $R(\mathrm{Ind})$ are defined as follows:

$$P(\mathrm{Ind}) := \frac{|\mathcal{H}(\mathrm{Ind}) \cap \mathcal{T}^+|}{|\mathcal{H}(\mathrm{Ind})|}, \qquad R(\mathrm{Ind}) := \frac{|\mathcal{H}(\mathrm{Ind}) \cap \mathcal{T}^+|}{|\mathcal{T}^+|}.$$

Then, the fitness $\mathrm{Fit}_\beta(\mathrm{Ind})$ with respect to a weighting parameter β is defined by the weighted F-measure

$$\mathrm{Fit}_\beta(\mathrm{Ind}) := \frac{(1+\beta^2)\cdot(P(\mathrm{Ind})\cdot R(\mathrm{Ind}))}{(\beta^2\cdot P(\mathrm{Ind}) + R(\mathrm{Ind}))}.$$

In view of our annotation application, we want to avoid false negatives in the preprocessing step possibly at the cost of precision. We therefore stress the recall value by setting $\beta = 2$. Here, note that even in case of a low precision in the keyframe-based preprocessing step, the subsequent MT-based annotation step may eliminate most of the false positives.

For the start of the optimization, we generate an initial population Π^0 consisting of M of individuals. To this end, we first compute a quantized motion template $X \in \{0,1,*\}^{f\times K}$ from $\mathcal{T}^+$ for a motion class $\mathcal{C}$ using a strict quantization threshold $\delta = 0$, see Sect. 3. Recall that X reveals the consistent aspects of the example motions and expresses characteristic properties of the class $\mathcal{C}$. However, using the columns of X directly as keyframes does not account for the negative training examples in $\mathcal{T}^-$. Our idea is to use the motion template only for the initialization and then successively refine the keyframes. To this end, for each of the M initial individuals we first choose a number N of keyframes using a normally distributed random number generator with a mean of 3. Then, to define a keyframe query $(\mathbf{V},\mathbf{d})$, we randomly pick N columns of X to define the keyframes $V_1,\ldots,V_N$. The distance parameters are initialized based on the distances of the chosen keyframes admitting some randomly chosen tolerance.

After the initialization, the three genetic operations referred to as *selection*, *recombination*, and *mutation* are used to iteratively breed a new population from a given population. Let Π^g, $g \in \mathbb{N}_0$, denote the current population. Then, using the concept of universial stochastic sampling, we select r individual from Π^g, which are referred to as *parents*. In the recombination step, the keyframes of any two of these parents are combined to derive new individuals, referred to as *offsprings*. To this end, we randomly chose a number of keyframes of each of the two parents and merge these keyframes to from a single keyframe sequence. The novel distance parameters are determined similarly to the initialization step. To avoid an early convergence of the optimization procedure towards a poor local optimum, one additional modifies the offsprings by suitable random operations referred to as mutations. In our case, an offspring is mutated by randomly choosing and applying one of the following operations:

- Add or remove a randomly chosen keyframe.
- Specialize (i.e., change $*$ to 0 or 1) or generalize (i.e., change a value 0 or 1 to $*$) a randomly chosen keyframe.
- Randomly increase and decrease the values in $\mathbf{d}$.

After the recombination and mutation step, we obtain $\frac{r(r-1)}{2}$ offsprings. We arrange the M individuals of Π^g and the $\frac{r(r-1)}{2}$ offsprings in a sorted list with decreasing fitness. Finally, the new population Π^{g+1} is obtained by picking the M fittest individuals from this list. This entire procedure is iterated for $g = 1,\ldots,G$, where G denotes a fixed number of generations. The fittest individual of Π^G is the solution of the optimization procedure.

In our implementation the population size is set to $M = 50$, the number of parents to $r = 5$, and the number of generations to $G = 100$. The exact values of these parameters, which have been determined experimentally, are not of crucial importance for the final result. However, as typical for evolutionary algorithms, different runs of the overall procedure may result in significant differences between the keyframes of the various solutions. Therefore, for each motion class, we run the overall genetic algorithm several times (in our experiments 100-500 times) and then pick an individual with keyframes that most frequently occur in the solutions. Further implementation details and running times of the genetic algorithm are discussed in Sect. 7.

7. Experiments

We implemented the learning and annotation algorithms in Matlab while passing time critical parts to subroutines implemented in C/C++. The computations were performed on an AMD Athlon X2 5000+ with 3.5 GB of RAM. For our

Class ID	class	description
$\mathcal{C}_1$	neutral	stand in a neutral position, hands lowered
$\mathcal{C}_2$	tpose	stand in t-pose, hands horizontally extended
$\mathcal{C}_3$	move	2 steps (starting left or right, walk, jog, run, ...)
$\mathcal{C}_4$	turn	turn around left or right
$\mathcal{C}_5$	sitLieDown	sit down on chair or floor, kneel, lie down on floor
$\mathcal{C}_6$	standUp	stand up from chair or floor
$\mathcal{C}_7$	hopOneLeg	jump with left or right leg
$\mathcal{C}_8$	jump	jump with both feet, jumping jack
$\mathcal{C}_9$	kick	kick to front or side with left or right leg
$\mathcal{C}_{10}$	punch	punch to front or side with left or right hand
$\mathcal{C}_{11}$	rotateArms	rotate both or single arms front or back
$\mathcal{C}_{12}$	throwR	throw an item with right hand, sitting or standing
$\mathcal{C}_{13}$	grabDepR	grab or deposit with right arm high, middle, low
$\mathcal{C}_{14}$	cartwheeel	cartwheel with left or right hand starting
$\mathcal{C}_{15}$	exercise	elbow to knee, skier, squat

Table 1: *The* 15 *motion classes used in our experiments.*

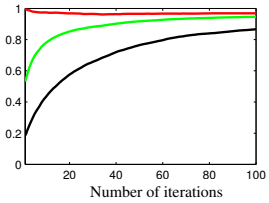

Figure 5: *Average precision (black), recall (red), and fitness (green) of the learnt keyframe queries evaluated on the training data as a function of the number of iterations used in the genetic algorithm.*

experiments, we assembled an evaluation dataset consisting of 109 mocap documents having an average length of 40 seconds each. The total length amounts to roughly 74 minutes (133019 frames at 30 Hz). To illustrate the scalability of our annotation procedure, we used mocap data from two different sources: 60 minutes where drawn from the HDM05 database [MRC*07] and 14 minutes from the CMU database [CMU03]. We manually annotated all 109 documents on the subsegment level according to the 15 classes described in Table 1. These classes were assembled with respect to the actions performed in the HDM05 motion database. To illustrate the practicability of our annotation procedure, we used various kinds of classes including rather general motion classes such as 'move', more specialized classes such as 'cartwheel', and rather uncharacteristic classes such as 'grabDepR'. Here, the more general classes are assembled from various subclasses. For instance, four different subclasses (sit down on chair or floor, kneel, lie down on floor) contribute to the class 'sitLieDown'. To obtain the annotations on the class level, one can simply combine the annotations on the subclass level. At this point, we emphasize that the particular choice of the motion classes is not of crucial importance. The choice was driven by the availability of the mocap data and by our motivation to give a comprehensive demonstration of the algorithms' performance (even in the presence of more critical classes such as 'grabDepR'). The concepts presented in this paper are generic in the sense that the underlying set of motion classes may easily be extended or modified to satisfy a user's specific needs.

Prior to the actual annotation step, we need to learn the motion templates and keyframe queries for each of the classes $\mathcal{C}_p$, $p \in [1:P]$. To this end, we assembled a training database of 24 minutes total length (42586 frames), which consists of nine example motions for each class, serving as $\mathcal{T}_p^+$, respectively. These example motions were manually cut out from additional HDM05 documents that are disjoint to all evaluation documents. In a first step, the relational features, which are needed for learning the motion templates as well as the keyframe queries, are computed and stored for the entire training examples (taking 137 seconds for the 24 minutes of data). From the features, we computed the quantized class motion templates using an iterative warping and averaging algorithm (see Sect. 3), which took roughly 3 seconds on average for each MT. To learn the keyframe queries, we also need negative example motions for each class. Here, we simply define $\mathcal{T}_p^-$ to be the union of all example motions that do not belong to the class $\mathcal{C}_p$: $\mathcal{T}_p^- = \bigcup_{q \in [1:P] \setminus p} \mathcal{T}_q^+$. Applying the genetic operations in an iterative fashion leads to significant improvements of the keyframes. As illustration we refer to Fig. 5, which shows the discriminative of the learnt keyframes over the iterations in terms of average precision, recall, and fitness (using Fit_2) on the training data. Here, averages are taken over the individuals of a population and over all motion classes. Remember that for the initialization of the keyframe queries, we pick columns from a quantized MT. Using a quantization threshold $\delta = 0$ (being very strict to variations in the training data), this quantized MT typically contains many wildcard characters. On the one hand, the so-chosen queries have a recall close to one on the training data, but on the other hand, the discriminative power against other classes is low, yielding a small precision value. We chose this strategy to steer the generated keyframe queries to a high recall with the goal to avoid false negative annotations in the preprocessing step. During the iterations, keyframe queries are refined and tuned towards a higher fitness. As seen in Fig. 5, a strong increase in the precision leads to the improvement in the fitness of the queries, at cost of a small decrease in recall.

Using the genetic algorithm with the parameters as specified in Sect. 6, it took roughly 10 seconds on average to learn a keyframe query for a given motion class. Since we run the overall genetic algorithm several times (100-500 times) to derive more characteristic keyframes, running time increases by a corresponding factor. Recall that for computing the fitness of an individual, one needs to perform a keyframe-based retrieval on the 24-minute training database. On average, the retrieval time was roughly 3 milliseconds. This operation has to be performed several thousand times for each run of the genetic algorithm. After computing the motion

templates and keyframe queries, these class representations are stored on hard disc for later usage.

Having completed the preprocessing step, our annotation procedure allows for efficiently annotating arbitrary and large sets of unknown mocap documents according to the given set of motion classes (or subsets thereof). To automatically annotate our evaluation database (109 documents, total length of 74 minutes), we proceed as follows. First, we extract the relational features and index the mocap documents using a standard inverted file index [MRC05]. In our implementation, the feature extraction takes roughly 250 seconds, whereas the indexing takes 4 seconds. Using the purely MT-based annotation procedure as described in Sect. 4, it takes 305 seconds to annotate the 109 documents (here, the index structure is not needed). Applying the keyframe-based preselection (Sect. 5), the running time of the overall annotation procedure decreases to 20 seconds, amounting to a speedup factor of more than 15. Here, processing a single keyframe query on the 74-minute evaluation database takes on average only 4 milliseconds (using the index structure), which is negligible compared to the MT-based annotation step.

The keyframe-based preselection step not only yields a significant speedup of the overall annotation procedure, but also has a considerable impact on the final annotation quality. This is affirmed by our experiments, where we qualitatively evaluated various variants of our annotation procedure. To this end, we compared the automatically generated annotations with manually generated ground truth annotations by means of two different performance measures. As first measure, we consider precision and recall values on the *frame level*. More precisely, for a given mocap document D of length L we define the sets

$$M(D) := \{(\ell,p)\,|\,\text{frame } \ell \text{ manually annotated by } p\} \text{ and} \quad (5)$$

$$A(D) := \{(\ell,p)\,|\,\text{frame } \ell \text{ automatically annotated by } p\}, \quad (6)$$

where $(\ell,p) \in [1:L] \times [1:P]$. In other words, the set $M(D)$ describes the manually generated or *relevant* annotations, whereas the set $A(D)$ describes the automatically generated or *retrieved* annotations produced by our procedure. Then, precision and recall of our annotation procedure are expressed by

$$P_1(D) := \frac{|M(D)\cap A(D)|}{|A(D)|} \text{ and } R_1(D) := \frac{|M(D)\cap A(D)|}{|M(D)|}. \quad (7)$$

Furthermore, let

$$F_1(D) := \frac{2P_1(D)R_1(D)}{P_1(D)+R_1(D)} \quad (8)$$

be the resulting F-measure. Note that $P_1(D) = 1$ in case of all retrieved annotations being among the relevant annotations (no "false positive"), whereas $R_1(D) = 1$ in case of all relevant annotations being retrieved. The frame-based performance measure F_1 may be problematic, since the beginning and ending of a motion of a specific class is often ambiguous. For example, consider a mocap document showing a person who sits down on a chair and remains seated for a long time. Then, it is not clear where exactly to set the end frame when manually annotating the document with respect to the class 'sitDownChair'. Also certain motion transitions from one class to another (e.g., from 'move' to 'turn') can often not be exactly specified. To account for such ambiguities, we use a second performance measure by considering precision and recall on the *segment level*. Here, we only check for overlaps of a manually annotated motion segment and an automatically generated segment both bearing the same class label p. We then define the segment-based precision $P_2(D)$, recall $R_2(D)$, and F-measure $F_2(D)$ analogously to the frame-based case. Note that the segment-based measures are more tolerant to smaller deviations in the annotations than the relatively strict frame-base measures. Therefore, the actual annotation quality is described well by the range defined by the values $F_1(D)$ and $F_2(D)$.

		P_1	R_1	F_1	P_2	R_2	F_2
total	without keyframes	0.48	0.78	0.60	0.57	0.91	0.70
	with keyframes	**0.69**	**0.79**	**0.74**	**0.78**	**0.88**	**0.82**
HDM	without keyframes	0.49	0.80	0.61	0.61	0.91	0.73
	with keyframes	**0.70**	**0.80**	**0.75**	**0.80**	**0.88**	**0.83**
CMU	without keyframes	0.41	0.75	0.53	0.39	0.90	0.54
	with keyframes	**0.66**	**0.74**	**0.70**	**0.65**	**0.91**	**0.76**

Table 2: *Various performance measures for our MT-based annotation procedure without and with keyframe-based preselection.*

To compute the performance measures on the entire evaluation database, we simply concatenated the 109 documents to form a single document and applied the above calculation steps, where we performed our annotation procedure without as well as with the keyframe-based preselection step. The results are shown in Table 2. For example, the precision P_1 without using keyframes is 0.48 and increases significantly to 0.69 when using our automatically computed keyframe queries. At the same time, the recall R_1 slightly increases from 0.78 to 0.79. While the increase in precision is expected when using keyframes, the increase in recall is somewhat surprising at first sight. Here, one reason is that by eliminating false positives, some of the relevant annotations that have previously been "overlayed" by false positive annotations emerge when using our minimization strategy, see Eq. (3). This again demonstrates that the keyframe-based preselection step eliminates a large number of false positive annotations while not loosing (or even yielding) relevant annotations. Fig. 6 shows some representative examples For example, note that the false positive annotations from the rather unspecific class 'grabDepR' could be eliminated using the keyframes. Next, consider the segment between frames 50 and 150 in Fig. 6 (b). Here, the actor shouts out having both hands raised in front of the mouth. As this motion is not related to any of the employed 15 classes, no manual annotation has been generated for these frames. Without using keyframes, our automatic procedure considers them most similar to either a 'throwR' or a 'grabDepR' motion. Using keyframes as hard constraints, these false positives are

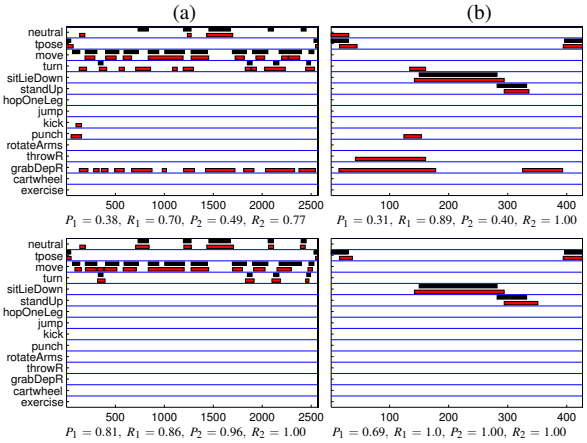

Figure 6: *Influence of the use of keyframes on the overall annotation result. Upper row: annotation result without keyframes. Lower row: annotation result with keyframes.*

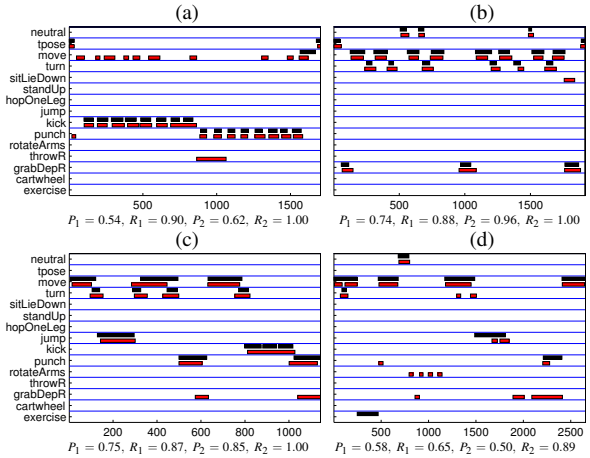

Figure 7: *Representative annotation results for two HDM05 ((**a**),(**b**)) and two CMU ((**c**),(**d**)) documents.*

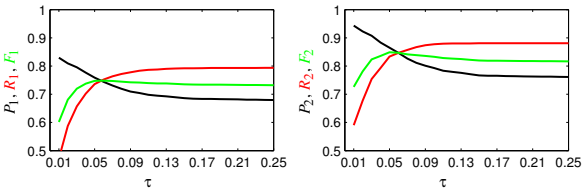

Figure 8: *Impact of the quality threshold τ on the frame-based (left) and segment-based (right) performance measures (using the annotation procedure with keyframe-based preselection). Black: precision. Red: recall. Green: F-measure.*

eliminated. As a consequence, the precision values receive a significant boost. In the accompanied video, we show animated videos of the presented annotation results along with the performed actions.

As expected, the segment-based precision and recall values are higher than the frame-based values, see Table 2. For example, using keyframes, one has $P_2 = 0.78$ (opposed to $P_1 = 0.69$). In other words, only 22% of the retrieved annotated segments are false positives. For the segment-based recall, one obtains $R_2 = 0.88$ (opposed to $R_1 = 0.79$). Here, only 12% of the relevant annotations are missing. Note that the frame-based performance measures are generally too strict whereas the segment-based ones are generally too tolerant. So, in summary, the actual performance of our overall annotation procedure can be described by the two F-measures $F_1 = 0.74$ (being pessimistic) and $F_2 = 0.82$ (being optimistic).

As was mentioned above, the HDM05 mocap data used for training is not contained in the evaluation data. However, the various motions corresponding to a specific class, even though performed by various actors executed with significant variations, are still somewhat controlled by general performance specifications. We therefore also evaluated our procedure on CMU documents containing at least some subsegments corresponding to our 15 classes. Table 2 shows the various performance measures separately for the HDM05 and CMU documents. Due to significant motion variations in the CMU data, some of which are not well reflected by the HDM05 training material, one has a decline in performance. For example, the F-measures of our overall procedure for the CMU data ($F_1 = 0.70$, $F_2 = 0.76$) are a bit lower than for the HDM05 data ($F_1 = 0.75$, $F_2 = 0.83$.

Fig. 7 depicts representative annotation results for both HDM05 and CMU documents. The annotation of a more problematic CMU document is shown in Fig. 7 (d). Here, the motion segment around frame 1000 erroneously received the annotation 'rotateArms'. A manual inspection shows that this segment actually consists of several arm swings—a motion type that is not reflected in the 15 motion classges used for the annotation. Furthermore, an exercise motion (around frame 400) was not annotated. Here, it turned out that the motion did not satisfy the keyframe constraints learned from HDM05 data. The performed actions can be reviewed in the accompanied video. On our project homepage http://www.mpi-inf.mpg.de/resources/MocapAnnotation we show videos along with the manual and automatic annotations of all 109 evaluation documents.

In all of the above experiments, we used the quality threshold $\tau = 0.13$. Actually, the choice of τ influences the quality of the overall annotation result. Note that a small value of τ poses a stronger condition on what to consider similar, thus leading to higher precision and lower recall, while a large value of τ has the opposite effect. To find a good trade-off of having high precision as well as high recall,

we computed the various performance measures for different values of τ, see Fig. 8. Our final choice of $\tau = 0.13$ is motivated by the request of having high recall values possibly at the expense of some additional false positive annotations.

8. Conclusions

In this paper, we presented a robust and efficient procedure for annotating large collections of unknown motion capture material. Using motion templates, we are able to identify logically related motions even in the presence of significant numerical differences. Using keyframe queries, we are able to efficiently prune the search space and to eliminate false positives. As a further contribution, we showed how characteristic keyframes can be learned from positive and negative training motions using a genetic algorithm. We reported on various experiments to demonstrate the practicability of our annotation procedure. Our concept is generic in the sense that it allows a user to easily adapt and modify the annotation types simply by exchanging the underlying motion classes. Because of their explicit semantic interpretation, even a manual design or tuning of motion templates and keyframes is feasible in case no suitable example motions are available. For the future, we plan to apply our annotation procedure to efficiently generate suitable prior knowledge as needed to stabilize and support human motion tracking [BRCS06]. Another application we have in mind is to apply our concept for automatically annotating various types of gesture as needed for generating gesture animations for novel text [NKAS08].

Acknowledgements: We thank Bernd Eberhardt from HDM Stuttgart for providing us with the HDM05 mocap data [MRC*07]. The CMU data was obtained from [CMU03] (funded by NSF EIA-0196217). The first author is funded by the Cluster of Excellence on Multimodal Computing and Interaction and the second author by the German Research Foundation (DFG CL 64/5-1).

References

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Trans. Graph. 22*, 3 (2003), 402–408.

[BH00] BRAND M., HERTZMANN A.: Style machines. In *Proc. ACM SIGGRAPH 2000* (2000), Computer Graphics Proc., ACM Press, pp. 183–192.

[BMS08] BAAK A., MÜLLER M., SEIDEL H.-P.: An efficient algorithm for keyframe-based motion retrieval in the presence of temporal deformations. In *Proc. ACM Int. Conf. on Multimedia Information Retrieval, Vancouver, Canada* (2008).

[BRCS06] BROX T., ROSENHAHN B., CREMERS D., SEIDEL H.-P.: Nonparametric density estimation with adaptive anisotropic kernels for human motion tracking. In *Proc. 2nd Workshop on Human Motion* (2006), vol. 4814 of *LNCS*, Springer, pp. 152–165.

[BSP*04] BARBIC J., SAFONOVA A., PAN J.-Y., FALOUTSOS C., HODGINS J. K., POLLARD N. S.: Segmenting motion capture data into distinct behaviors. In *GI '04: Proc. Graphics interface* (2004), Canadian Human-Computer Communications Society, pp. 185–194.

[CH05] CHAI J., HODGINS J. K.: Performance animation from low-dimensional control signals. *ACM Trans. Graph. 24*, 3 (2005), 686–696.

[CHP07] COOPER S., HERTZMANN A., POPOVIĆ Z.: Active learning for real-time motion controllers. *ACM Trans. Graph. 26*, 3 (2007), 5.

[CMU03] CMU: Carnegie-Mellon Mocap Database. http://mocap.cs.cmu.edu, 2003.

[FF05] FORBES K., FIUME E.: An efficient search algorithm for motion data using weighted PCA. In *Proc. 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2005)* (2005), ACM Press, pp. 67–76.

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph. 23*, 3 (2004), 559–568.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Trans. Graph. 21*, 3 (2002), 473–482.

[KPZ*04] KEOGH E. J., PALPANAS T., ZORDAN V. B., GUNOPULOS D., CARDLE M.: Indexing large human-motion databases. In *Proc. 30th VLDB Conf., Toronto* (2004), pp. 780–791.

[LZWM05] LIU G., ZHANG J., WANG W., MCMILLAN L.: A system for analyzing and indexing human-motion databases. In *Proc. 2005 ACM SIGMOD Intl. Conf. on Management of Data* (2005), ACM Press, pp. 924–926.

[MR06] MÜLLER M., RÖDER T.: Motion templates for automatic classification and retrieval of motion capture data. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), ACM Press, pp. 137–146.

[MRC05] MÜLLER M., RÖDER T., CLAUSEN M.: Efficient content-based retrieval of motion capture data. *ACM Trans. Graph. 24*, 3 (2005), 677–685.

[MRC*07] MÜLLER M., RÖDER T., CLAUSEN M., EBERHARDT B., KRÜGER B., WEBER A.: *Documentation: Mocap Database HDM05*. Computer Graphics Technical Report CG-2007-2, Universität Bonn, June 2007. http://www.mpi-inf.mpg.de/resources/HDM05.

[Mül07] MÜLLER M.: *Information Retrieval for Music and Motion*. Springer, 2007.

[NKAS08] NEFF M., KIPP M., ALBRECHT I., SEIDEL H.-P.: Gesture modeling and animation based on a probabilistic recreation of speaker style. *ACM Trans. Graph. 27*, 1 (2008).

[PB02] PULLEN K., BREGLER C.: Motion capture assisted animation: Texturing and synthesis. In *Proc. SIGGRAPH 2002* (2002), ACM Press, pp. 501–508.

[Poh99] POHLHEIM H.: *Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise*. Springer, 1999.

[RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: multidimensional motion interpolation. *IEEE Comput. Graph. Appl. 18*, 5 (1998), 32–40.

[WCYL03] WU M.-Y., CHAO S., YANG S., LIN H.: Content-based retrieval for human motion data. In *16th IPPR Conf. on Computer Vision, Graphics and Image Processing* (2003), pp. 605–612.