# Efficient Motion Retrieval in Large Motion Databases

Mubbasir Kapadia* I-kao Chiang† Tiju Thomas‡ Norman I. Badler§ Joseph T. Kider Jr¶

University of Pennsylvania

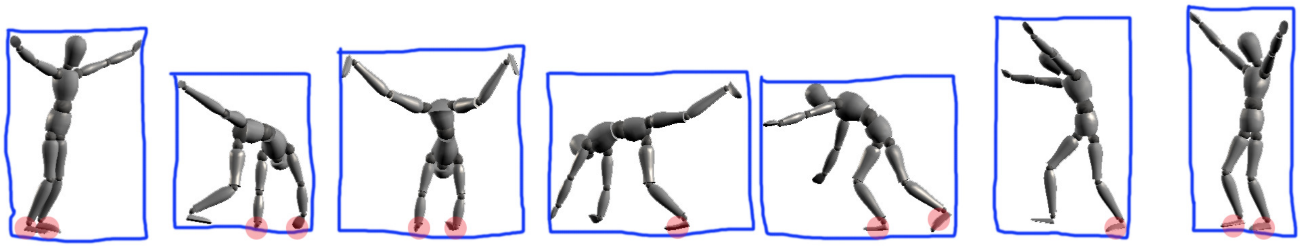

**Figure 1:** *Combination of Support Key and Shape Key (specified using sketches) used for retrieving a cart-wheel motion in the CMU motion database.*

## Abstract

There has been a recent paradigm shift in the computer animation industry with an increasing use of pre-recorded motion for animating virtual characters. A fundamental requirement to using motion capture data is an efficient method for indexing and retrieving motions. In this paper, we propose a flexible, efficient method for searching arbitrarily complex motions in large motion databases. Motions are encoded using *keys* which represent a wide array of structural, geometric and, dynamic features of human motion. Keys provide a representative search space for indexing motions and users can specify sequences of key values as well as multiple combination of key sequences to search for complex motions. We use a trie-based data structure to provide an efficient mapping from key sequences to motions. The search times (even on a single CPU) are very fast, opening the possibility of using large motion data sets in real-time applications.

**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, laban movement analysis, indexing, retrieval

*e-mail: mubbasir.kapadia@gmail.com
†e-mail:igorchiang@gmail.com
‡e-mail:tiju@seas.upenn.edu
§e-mail:badler@seas.upenn.edu
¶e-mail:kiderj@seas.upenn.edu

## 1 Introduction

In the last decade, the games and visual effects industry has experienced a paradigm shift from procedural methods to using pre-recorded motion for animating virtual human characters. This is due to the increasing availability of large motion databases [CMU 2003; UTA 2011] and recent advancements in motion re-targeting [Gleicher 1998] and motion synthesis [Kovar et al. 2002; Arikan et al. 2003]. Efficient indexing and retrieval of motions is a precursor to using motion capture data. The human anatomy has a complex skeletal structure with joints connected at multiple levels of hierarchy. The space of human pose configurations is continuous, non-linear, and extremely high dimensional which makes searching in this space prohibitive. In addition, there are many features of motion [Laban 1971] including structural, geometric, dynamic, and even emotional features that we may want to query. Hence, there is a growing need to define an efficient, yet representative encoding of human motion data for indexing and retrieval.

Prior work in motion retrieval is classified based on choice of encoding, query representation, and measure of similarity. Text-based indexing requires manual labeling of motion clips, which is not universal and may not capture the nuances of each motion. Motion clips as queries provide the most direct mapping to search motion data. Numerical-based retrieval methods [Kovar and Gleicher 2004] return structurally similar results, but cannot effectively capture logically and contextually similar motions (e.g., two jump motions may have widely different pose transitions). Content-based retrieval methods [Müller et al. 2005; Müller and Röder 2006] define features as geometric relations between specific body joints, with feature selection greatly impacting performance and quality of results. Sketch-based interfaces [Chao et al. 2011; Choi et al. 2012] can be used to intuitively specify static poses but cannot capture the dynamic properties of motion. These methods are limited in search flexibility and ability to handle large databases. The far-reaching goal is to provide a compact representation that sufficiently captures the features of human motion and provides an efficient means of querying and indexing this search space independent of database size.

In this paper, we propose a flexible, efficient method for searching motions in very large databases. Motions are encoded using *keys* which represent the wide array of structural, geometric, and dynamic features of human motion, and can be extended to meet the needs of specific applications. Keys provide a representative search space for indexing motions and users can use sequence of key values as well as multiple combinations of keys to search for complex motions in the database. A prefix-tree based data structure provides an efficient mapping from "key sequences" to motions, with very fast search times (even on a single CPU) that is independent of the size of the database. This facilitates efficient indexing of arbitrarily large motion databases without compromising on query complexity, opening the possibility of using large motion data sets in interactive applications.

## 2 Related Work

**Motion Index Storage and Retrieval.** The work in [Kovar and Gleicher 2004] parameterizes motions by precomputing "match webs". Neigbhor graphs are proposed in [Chai and Hodgins 2005] for storage and indexing. However, the quadratic memory requirement in these methods makes it impractical for large databases. Content-based retrieval methods [Müller et al. 2005; Müller and Röder 2006] compute a small set of geometric properties which are used to find logically similar motions.

Following the pioneering work of [Faloutsos et al. 1994], different techniques [Keogh et al. 2004; Krüger et al. 2010] have been used for spatial indexing of motion data. The work in [Barbič et al. ; Liu et al. 2005] uses Principal Component Analysis to create a reduced linear representation of human motion which are used as indices for motion retrieval. The work in [Chao et al. 2011] uses a set of orthonormal spherical harmonic basis functions to represent motion trajectories. The work in [Deng et al. 2009; Wu et al. 2009] reduces the search space by clustering motion data based on body segments and uses string matching algorithms for efficient runtime query processing.

**Query Specification Interface.** Keyword based systems [CMU 2003; Mixamo 2010] are cost-efficient and widely used, but are not robust to language and semantics, and cannot capture the complexities in motion data. Sketch-based interfaces [Chao et al. 2011; Thorne et al. 2004; Choi et al. 2012; Li et al. 2006] provide a more intuitive means of query specification but cannot capture the dynamics of human motion. Performance capture devices such as robots [Numaguchi et al. 2011], dolls [Feng et al. 2008], as well as hybrid approaches [Lee et al. 2002] have also been used to search motions. Recent work [Tautges et al. 2011] proposes the use of acceleration information of end effectors captured using accelerometers for motion retrieval.

**Visualization.** The work in [Assa et al. ] generates image sequences to provide an "action synopsis" of the motion. Heat maps [Ren 2006] provide a useful indication of where the motion is present. Highlighting relevant keyframes [Fauvet and Bouthemy 2004; Xiao et al. 2006] is particularly useful to get a quick overview of large motions. Posture keys [Sakamoto et al. ] and motion cues [Bouvier-Zappa et al. 2007] provide helpful annotations to augment the visualization of the motion.

**Laban Movement Analysis.** Laban Movement Analysis (LMA) [Laban 1971; Bartenieff 1972; Maletic 1987] identifies features which describe the structural, geometric, and dynamic properties of human motion. LMA has been used for gesture animation [Zhao and Badler 2005], motion expression [Chi et al. ], motion segmentation [Bouchard and Badler 2007] and learning motion styles [Torresani et al. 2006]. The work in [Wakayama et al. 2010; Okajima et al. 2012; Yu et al. 2005] have demonstrated the use of a subset of LMA features for motion retrieval.

**Comparison to Prior Work.** Our work supplements excellent contributions in numerical-based [Kovar and Gleicher 2004] and

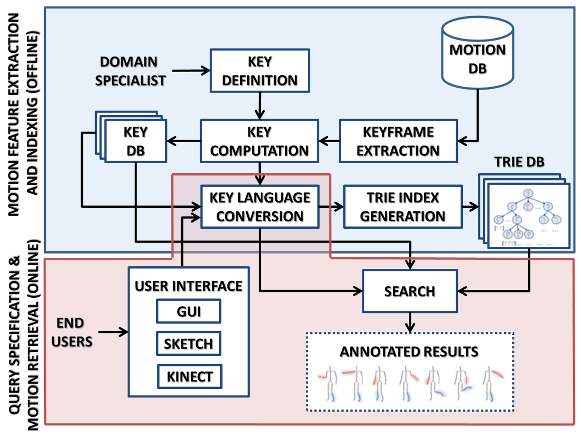

**Figure 2:** *Framework Overview.*

content-based [Müller et al. 2005; Müller and Röder 2006] retrieval methods to provide a rich set of structural, geometric, and dynamic features which can be used for efficiently indexing [Krüger et al. 2010; Wu et al. 2009] motion data. In particular, we are inspired by the work in [Wakayama et al. 2010; Okajima et al. 2012; Yu et al. 2005] and use the theories of Laban [Laban 1971] to define this feature set. In comparison to the binary features described in [Müller et al. 2005], motion keys can take any vocabulary to quantify higher-order motion features. To facilitate efficient indexing in this large space of motion keys, we present a method that supports fuzzy subsequence matching operations at interactive rates on large motion databases[CMU 2003].

## 3 Framework Overview

Figure 2 presents an overview of our framework. Our method takes as input raw motion data and extracts meaningful features from the motion to provide a compact, representative space to index into the database. End users specify queries as combinations of sequences of a variety of motion features and our framework returns motion subsequences that satisfy these properties.

**Motion Feature Extraction and Indexing (Offline).** Given a large motion database we first compress the motion data by extracting only relevant keyframes using the method described in [Xiao et al. 2006]. The motion data is represented as a set of curves for each joint angle. A set of candidate keyframes are selected which map to the extreme points of these curves to produce a simplified representation which approximates the original curve within a reasonable error threshold. For more details, please refer to [2006].

Next, we define an extensible set of *motion keys* which characterize the different structural, dynamic, and geometric properties of the motion over a time window (Section 4). Keys can be computed directly from the motion, or computed using other keys. During an offline process, we compute all key values for all motions in the database. Key values may be of different data-types and may have arbitrary ranges. For intuitive query specification and efficient indexing, we first define a minimal alphabet for each key. Key values are converted into this language to populate a trie data structure which facilitates efficient motion subsequence matching which is independent of the number of motions in the database (Section 5).

**Query Specification and Motion Retrieval (Online).** Motion keys provide a powerful interface for indexing the motion database, allowing end-users to search for arbitrarily complex motions by specifying sequences of combinations of key values (Section 6). The time complexity for motion retrieval is $O(n)$ where $n$ is the number of keys used in the search query, and is independent of database size. Complex motions can be retrieved at interactive rates on very large motion databases using our method (Section 7).

## 4 Motion Feature Extraction

The input to the framework is a set of motions $\mathbb{M} = \{m_{\text{i}}(t)|\forall i\}$ where each motion clip $m_{\text{i}}(t) = \{m_{\text{i}}(t_{\text{j}})|\forall j \in \{1, 2...n\}\}$ represents the sequence of joint configurations of the body over a period of time. The pose of the character at any point in the motion is given by $\mathbf{P} \in \mathbb{R}^{3\times|J|}$ where $\mathbf{P}^j$ is the position of the $j^{th}$ joint of the body. To make the motion data more tractable, we first extract key-frames for each motion to represent its most salient properties. The resulting motion clip $\{m_{\text{i}}(u_{\text{k}})|\forall k \in \{1, 2...m\}\}\}$ has much fewer frames $(m << n)$, but still captures all the relevant information needed to perform effective motion indexing. The method of key-frame extraction that we use is described here [Xiao et al. 2006].

There are many different properties of human motion [Laban 1971] which include: (1) structural and physical characteristics of the human body, (2) motion dynamics and intent of motion, and (3) shape morphology of the body during the motion, A fundamental requirement for indexing motion databases is to provide a representative search space which captures all these properties and allows users to characterize complex motions using combinations of these properties.

We define a set of *motion keys* $\{K(u)\}$ where each key captures a particular feature of the motion. Keys can be computed directly from the motion data, or derived from other keys. Keys are extensible – users may define their own keys to meet the needs of their application. A key value $K(u_{\text{k}})$ for a motion state $m(u_{\text{k}})$ is defined as a function $f$ of the the current motion state of the character, and a set of preceding and succeeding motion states. This allows the key to capture dynamic properties of the motion that accrue over a window of frames. Keys can also be derived from other key values and computed as a function $g$ of the values of a set of keys $\{K^{\text{a}}(u)\}$ over a window $[k - \Delta, k + \Delta]$. For example, any second order keys such as velocity or acceleration are computed as a function of the keys used to store position and orientation information. Note that even though key values are computed for every key-frame of a motion, keys are not simply per-frame features but characterize different properties of motion that may accumulate over a window of frames.

$$\begin{aligned} K(u_{\text{k}}) &= f(\{m(u_{\text{k}})|k - \Delta \leq k \leq k + \Delta\}) \\ &= g(\{K^{\text{a}}(u_{\text{k}})|k - \Delta \leq k \leq k + \Delta, \forall a\}) \end{aligned} \quad (1)$$

We define a set of body segments $\mathbb{S}$ for which motion keys can be computed (Figure 3). Different motion keys can be computed for different subsets of these body segments. For example, a motion key which characterizes shape has meaning for the entire body but does not have meaning for an individual end effector. Motion keys which capture displacement and orientation are particularly useful for the hands and feet.

**Key Computation.** During an offline process, we compute all the keys for all the motions in the database $\mathbb{M}$. A specific key value $K^{\text{a}}_{\text{s}}(u_{\text{k}})$ represents the value of the property $a$, body segment $s$, at the motion state $m(u_{\text{k}})$. Based on the theories of [Laban 1971; Maletic 1987] we define a rich set of keys for motion indexing, described in the rest of this section.

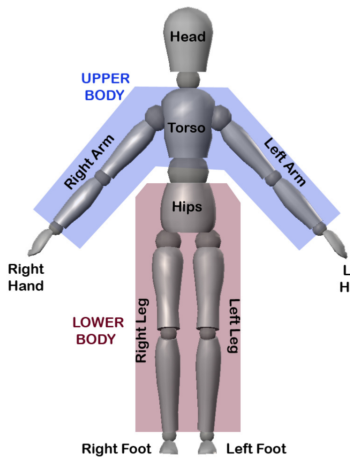

**Figure 3:** *Body Segments.*

### 4.1 Body Keys

Body keys characterize the structural and physical properties of human motion. This catagory helps annotate and identify which body part is moving, connection between different body parts, and patterns of movement between parts of the body.

**Action Presence.** This key denotes the presence or absence of movement in a specific body part over succesive frames and is computed by detecting if the displacement of the body segment between successive frames crosses a threshold.

$$K^{\text{action}}_{\text{s}}(u_{\text{k}}) = |\mathbf{P}^{\text{s}}(u_{\text{k}}) - \mathbf{P}^{\text{s}}(u_{\text{k-1}})| > \epsilon \;\; ? \quad 1:0 \qquad (2)$$

**Displacement and Orientation.** $K^{\text{disp}}_{\text{s}}(u_{\text{k}})$ calculates the displacement of an end effector $s$ (e.g., hand) from the root joint $r$ (e.g., shoulder) of its limb. Similarly, $K^{\text{orient}}_{\text{s}}(u_{\text{k}})$ computes the relative orientation of an end effector with respect to its root. $K^{\text{orient}}_{\text{head}}(u_{\text{k}})$ provides the current look-at direction of the character.

$$K^{\text{disp}}_{\text{s}}(u_{\text{k}}) = |\mathbf{P}^{\text{s}}(u_{\text{k}}) - \mathbf{P}^{\text{r}}(u_{\text{k}})| \qquad (3)$$

**Closest Body Segment.** $K^{\text{closest}}_{\text{s}}(u_{\text{k}})$ stores the index of the body segment that is in closest proximity to end effector $s$.

$$K^{\text{closest}}_{\text{s}}(u_{\text{k}}) = \arg\min_{i} |\mathbf{P}^{\text{s}}(u_{\text{k}}) - \mathbf{P}^{\text{i}}(u_{\text{k}})| \forall i \in \mathbb{S} \qquad (4)$$

**Closest Body Segment Distance.** $K^{\text{closest-dist}}_{\text{s}}(u_{\text{k}})$ computes the distance between the end effector and its closest body segment $K^{\text{closest}}_{\text{s}}(u_{\text{k}})$. These keys can be used to describe complex relations between body segments such as motion towards a part of the body or contact between two body parts.

$$K^{\text{closest-dist}}_{\text{s}}(u_{\text{k}}) = |\mathbf{P}^{\text{s}}(u_{\text{k}}) - \mathbf{P}^{\text{i}}(u_{\text{k}})| \text{ where } i = K^{\text{closest}}_{\text{s}}(u_{\text{k}}) \qquad (5)$$

**Center of Mass.** $K^{\text{com}}(u_{\text{k}})$ is computed as a weighted average of the positions of all joints in the body.

$$K^{\text{com}}(u_{\text{k}}) = \frac{\sum_{s\in\mathbb{S}} w_s \cdot \mathbf{P}^{\text{s}}(u_{\text{k}})}{\sum_{s\in\mathbb{S}} w_s} \quad (6)$$

**Center of Mass Displacement.** $K^{\text{com-disp}}(u_{\text{k}})$ computes the displacement of the center of mass of the body from its rest position. This key is particularly useful for specifying full-body postures where the character maybe crouching or jumping in the air.

$$K^{\text{com-disp}}(u_{\text{k}}) = |K^{\text{com}}(u_{\text{k}}) - K^{\text{com}}(u_{\text{rest}})| \quad (7)$$

**Balance.** $K^{\text{balance}}(u)$ is a boolean value to indicate the relative location of the center of mass with respect to the support polygon.

**Support.** $K^{\text{support}}(u)$ identifies the current body segment which is being used to support the body weight and is the body part which is in contact with the ground. The possible values for $K^{\text{support}}(u)$ are $\{$`LFoot`, `RFoot`, `BothFeet`, `LHand`, `RHand`, `BothHands`$\}$ and can be extended to include any body segment, if required.

### 4.2 Effort Keys

An angry or a gentle arm gesture are very similar in body structure, but differ greatly in the intent of motion. In order to capture the subtle dynamic characteristics of human motion, we define four effort keys, adapted from LMA: *space, weight, time, and flow*. Each key is computed using a dynamic feature in the motion and has two opposing polarities. Effort keys are computed for each key-frame in the motion; however, it captures the dynamic properties of motion that accumulate over a window of frames. The effort key value at a particular frame $u_k$ is the frame index $i < k$, which indicates that the motion in the interval $[i, k]$ satisfies the positive polarity for that particular effort key.

**Space.** The effort in space is used to characterize the spatial pattern of movement and may be *direct* or *indirect*. Indirect motion is interrupted and roundabout while direct motion proceeds along a mostly straight line without deviation. We use the ratio of displacement to net distance traveled to capture the space effort. $K_{\text{s}}^{\text{space}}(u_{\text{k}})$ stores the frame index $i$ such that the motion clip $\{m(u_{\text{i}}), m(u_{\text{i+1}})...m(u_{\text{k}})\}$ has direct space effort and is computed as follows:

$$K_{\text{s}}^{\text{space}}(u_{\text{k}}) = \arg \min_{i\in(0,k)} \frac{\sum_{j=i}^{k} |\mathbf{P}^{\text{s}}(u_k) - \mathbf{P}^{\text{s}}(u_j)|}{|\mathbf{P}^{\text{s}}(u_k) - \mathbf{P}^{\text{s}}(u_i)|} - \mathtt{T} \quad (8)$$

**Weight.** The effort of weight in motion can be light (e.g., a delicate touch) or it can be strong (e.g., a strong impact). The opposing polarities in weight effort are captured using deceleration of motion. Strong motion is characterized by large deceleration in motion of end effectors while light, fluid motion is representative of little or no deceleration. Note that weight effort is velocity independent. A running motion is strong because the footfalls or hand motions have peaks in deceleration at the end of a run cycle, not because the body moves quickly. Olympic walking, which is very fluid and fast, is not strong due to energy conservation, and has no peaks in deceleration.

$K_{\text{s}}^{\text{weight}}(u_{\text{k}})$ stores the frame index $i$ such that the motion clip $\{m(u_{\text{i}}), m(u_{\text{i+1}})...m(u_{\text{k}})\}$ has strong weight. For a particular frame $u_k$, $K_{\text{s}}^{\text{weight}}(u_{\text{k}}) = i$ where $i < k$ such that the total deceleration of the joint $s$ over the interval $[i, k]$ is nearly equal to a maximum threshold $T_{max}$, indicating strong weight.

$$K_{\text{s}}^{\text{weight}}(u_{\text{k}}) = \arg \min_{i\in[0,k)} \left( T_{max} - \frac{|\mathbf{v}^{\text{s}}(u_{\text{k}})| - |\mathbf{v}^{\text{s}}(u_{\text{i}})|}{u_k - u_i} \right) \quad (9)$$

**Time.** The effort in time captures the temporal alternation of movement. It may be sudden (e.g., repeated shaking of the fists in anger) or sustained. A sudden motion is characterized by peaks in acceleration while sustained motion has uniform velocity (no acceleration). The physical metric we use to quantiy time effort is the net acceleration accumulated over the duration of a motion interval. For a particular frame $u_k$, $K_{\text{s}}^{\text{time}}(u_{\text{k}}) = i$ where $i < k$ such that the net acceleration of the joint $s$ over the interval $[i, k]$ is approximately equal to a maximum threshold $T_{max}$, indicating that the motion clip $\{m(u_{\text{i}}), m(u_{\text{i+1}})...m(u_{\text{k}})\}$ has sudden motion.

$$K_{\text{s}}^{\text{time}}(u_{\text{k}}) = \arg \min_{i\in[0,k)} \left( T_{max} - \sum_{j=i+1}^{j\leq k} \frac{|\mathbf{v}^{\text{s}}(u_{\text{j}}) - \mathbf{v}^{\text{s}}(u_{\text{j-1}})|}{u_j - u_{j-1}} \right) \quad (10)$$

Acceleration is used as the physical quantity to compute both weight and time effort. Total deceleration over the motion clip quantifies weight effort while the aggregated acceleration over the motion is used to compute the time effort. This captures the symmetry as well as the subtle difference between the two effort quantities. For example, a repeated shaking of the fist is sudden and strong, while chopping wood (smooth repetitive motion with high peaks in deceleration) is sustained and strong.

**Flow.** Flow effort characterizes continuity in motion and is computed using the third-order physical metric, jerk. Bounded motion is extremely discontinuous with high jerk, whereas free motion has little or no change in acceleration.

$$K_{\text{s}}^{\text{flow}}(u_{\text{k}}) = \arg \min_{i\in[0,k)} \left( T_{max} - \sum_{j=i+1}^{j\leq k} \frac{|\mathbf{a}^{\text{s}}(u_{\text{j}}) - \mathbf{a}^{\text{s}}(u_{\text{j-1}})|}{u_j - u_{j-1}} \right) \quad (11)$$

### 4.3 Shape Keys

The shape that the body, or parts of the body make in any given pose, and the manner in which body shapes change is captured using shape keys. We compute shape keys $K_{\text{s}}^{\text{shape}}(u_{\text{k}})$ for each body part $s$ as follows:

$$K_{\text{s}}^{\text{shape}}(u_{\text{k}}) = \mathtt{BoundingVolume}(\{\mathbf{P}(u_{\text{k}})\}^s) \quad (12)$$

where $\mathtt{BoundingVolume}(\{\mathbf{P}(u_{\text{k}})\}^s)$ computes an axis-aligned bounding box that contains positions of all joints present in the body part $s$. $K_{\text{s}}^{\text{shape}}(u_{\text{k}})$ is computed for all body segments as well as the entire body.

## 5 Motion Indexing

The keys described in Section 4 provide a representative space for searching motions. In order to fully exploit the use of keys for motion retrieval, we must define an efficient mechanism for indexing the motion database which satisfies the following requirements:

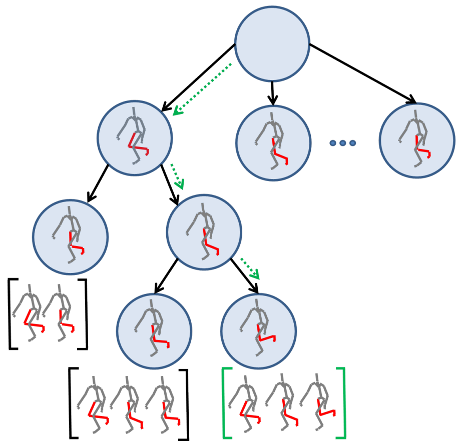

**Figure 4:** *Illustration of a trie used for motion key indexing. Each node in the trie corresponds to a specific value of the motion key (e.g., leg configuration). All leaf nodes contain a reference to the motion clip that matches the key value sequence by traversing the trie to reach the leaf node. A search for a particular sequence of key values returns the motion clips that are referenced by the leaf node descendents of the last node reached by traversing the trie along the sequence.*

- **Subsequence Matching.** A user must be able to specify queries of sequences of key values where search results may include smaller clips of motions within larger motions in the database.
- **Fuzzy Search.** Users may specify key values that need to be specified within a certain tolerance value. In addition, sequences of key values may not necessarily occur in successive key-frames of valid motion clips, as long as the ordering is maintained.
- **Online Motion Retrieval.** Search times must be efficient and independent of database size.

A trie [Fredkin 1960] is an $m$-ary tree used for efficient retrieval where the keys used for indexing are defined over a finite alphabet with a lexicographic ordering. The root node represents a null key, and each node in the trie corresponds to a specific instance of an alphabet in the key vocabulary. Hence, the maximum breadth of the trie is equal to the cardinality of the alphabet. Each node in the trie represents a prefix of one or more key instances, and all descendents of a node share a common prefix. A traversal from the root node to a leaf node in the trie corresponds to a specific key instance, and each leaf node can additionally store a reference to a *value* which the key may be used to index into. Tries support $O(1)$ find, insert and, delete operations for keys that can be represented by a unique string. The time complexity of a trie is independent of the number of elements in the dictionary and the number of elements in the key alphabet. The maximum depth of the trie is $l$ where $l$ is the length of the longest key in the dictionary; however, this can often be greatly reduced using memory optimizations where a node in the trie can map to a subsequence of alphabets in the key. The worst case complexity is $O(l)$, corresponding to a traversal to the greatest depth in the tree. Tries do incur an additional memory overhead; however, the additional memory requirement is mitigated for very large databases. We refer the reader to relevant literature [Fredkin 1960] describing the trie data structure and present an overview of its use for indexing into motion databases.

**Motion Retrieval using Tries.** Each motion key has a finite set of possible values it can take and there exists an ordering of key values. The domain of key alphabets are unsigned integers and each alphabet is defined using $\{\texttt{min}, \texttt{max}, \texttt{inc}\}$ where $\{\ \texttt{min}, \texttt{max}\ \}$ defines the range of the alphabet and $\texttt{inc}$ defines the resolution of discretization. Choosing an appropriate discretization provides a balance in flexibility of query specification, while generating an efficient, yet representative space for indexing. Figure 4 illustrates a motion key indexed using tries.

We use motion keys as an indexing mechanism by generating a trie for each computed motion key. Each leaf node in the trie maps to a unique sequence of key values for a particular key and contains a reference to the corresponding motion in the database. While the trie operations are independent of the number of elements in the alphabet, having a large vocabulary increases the sparsity of the trie where the trie has very little branching. This increases the memory footprint of the trie and also narrows the search results so that each motion satisfies a very specific sequence of key values. Hence, we define a minimal vocabulary for each key by calculating the minimum and maximum value that the key takes across the entire database of motions and choosing an appropriate discretization (increment) which differentiates motions with differing features while still clustering similar motions together with the same key value.

**Searching for motion subsequences.** An important requirement in motion indexing is the ability to retrieve motion subsequences that are contained in larger motion clips. This is achieved by returning all leaf nodes (and their associated motions) in a trie for which the traversal contains the subseqeunce of key values that is specified. A key subsequence may begin at any level in the trie. Hence, a subsequence is searched by first searching for occurrences of the first key value at all levels in the trie and then starting the traversal from that level to see if the sequence of key values is present. If the sequence ends before the frontier of the trie is reached, all leaf node descendents of the last node reached are returned as valid results. Due to the lexicographic ordering of key values, nodes at each level in the trie are stored in an ordered fashion. This allows us to efficiently search for the presence of a particular key value at each level. Equations 16 and 17 are examples of queries that search for motion subsequences.

**Fuzzy Motion Search.** A user can specify query with a tolerance value for a particular key value to indicate an approximate search where all motions that satisfy the key value within the tolerance range are valid results. The ordering of key values at each level allows us to quickly search for nodes whose key values are within the tolerance range. The remainder of the query sequence is then independently searched starting from each of these nodes, with the final result equal to the union of each of these searches. Equation 14 is an example of a fuzzy search query.

## 6 Motion Retrieval

**Input Query.** The motion keys $\{K^{\mathrm{a}}(u) \forall a\}$, described in Section 4 provide a representative space for describing complex motion queries. Each key has an alphabet, and we can specify an atomic query as a regular expression on a single key alphabet. Regular expressions provide a mechanism to specify orderings of key values. In addition, a user may also specify strict temporal constraints that a particular key value must be specified at a given point of time (or two different keys must satisfy constraints at the same point of time). By using a logical combination of atomic queries on different keys, complex motions can be easily described. A composite query $Q$ is specified as follows:

$$Q = \coprod_{a \in A} \mathrm{q}(K^{\mathrm{a}}(u)) \tag{13}$$

where $\coprod$ denotes the logical and-or combination of multiple atomic queries for a set of motion keys $\{K^{\mathrm{a}}(u)|a \in A\}$. Each atomic query $\mathtt{q}(K^{\mathrm{a}}(u))$ is a regular expression describing a specific ordering of key values, in addition to providing temporal constraints. The following are sample atomic queries that can be specified for a motion key:

$$\mathtt{q}_1(K^{\mathrm{a}}(u)) \leftarrow (k^1, \epsilon) \cdot (k^2) \tag{14}$$
$$\mathtt{q}_2(K^{\mathrm{a}}(u)) \leftarrow (k^1_{u=u_1}) \cdot (k^2_{u=u_2}) \tag{15}$$
$$\mathtt{q}_3(K^{\mathrm{a}}(u)) \leftarrow (-)^* \cdot (k^1) \cdot (-)^* \tag{16}$$
$$\mathtt{q}_4(K^{\mathrm{a}}(u)) \leftarrow (-)^+ \cdot (k^1) \cdot (-)^+ \cdot (k^2) \tag{17}$$

$\mathtt{q}_1$ describes the occurrence of two key values $k^1$ and $k^2$ one after another where the value of $k^1$ may be satisfied within a tolerance range $\epsilon$. $\mathtt{q}_2$ describes the occurrence of two key values $k^1$ and $k^2$ at specific points in the motion. $\mathtt{q}_3$ describes the presence of a key value somewhere in the motion with zero or more key-frames present before and after it. $\mathtt{q}_4$ describes the presence of a key value somewhere in the motion with one or more key-frames preceding and succeeding it. Using these simple constructs, more complex queries can be defined. In addition, a user may also specify temporal constraints across multiple keys (i.e., two keys must satisfy a key value at the same point of time in the motion).

**Search Results for Atomic Queries.** A regular expression on each key $K^{\mathrm{a}}(u)$ maps to an independent search on the trie for that particular key. The output of each search is a set of resulting motion clips $\{R^a\}$:

$$\textbf{Search}(\mathtt{q}(K^{\mathrm{a}}(u))) \rightarrow \{R^a | R^a = \{\mathtt{i}, u_s, u_e, [u]\}\} \tag{18}$$

where $\mathtt{i}$ is the motion index, $\{u_s, u_e\}$ is the start and end of the motion clip and, $[u]$ are the frame indices when the specific key values were satisfied.

**Temporal Constraints for Atomic Queries.** If temporal constraints for a particular key are specified, the search result $\{R^a\}$ is filtered to remove motion clips which don't satisfy the constraint. This process is linear in the number of search results.

**Search Result for Composite Query** Based on the logical combination of atomic queries, the result sets of each atomic query are combined as follows:

$$R = \biguplus_{a \in A} \{R^a\} \tag{19}$$

where $\uplus$ is the set operation $\{\bigcap, \bigcup\}$ depending on the logical combination between the atomic queries. Two results are considered to be the same if and only if the motion index $\mathtt{i}$ is the same, $\{u_s, u_e\}$ in both motion clips are within some threshold of each other and, $[u]$ satisfies any temporal constraints that were placed across multiple keys.

## 7 Results

We demonstrate the effectiveness of our approach by searching for complex motions in the CMU [CMU 2003] and Texas HMD [UTA 2011] motion databases. The two databases combined have 6121 animations, and a total of 35 hours of motion capture data. The animation length ranges from 0.5 to 96 seconds, with many animations lasting thousands of frames. Our system uses a total of 43 motion keys to index into the database capturing a wide range of motion features. The memory footprint of each motion key is 30 MB (linear in number of extracted key frames).

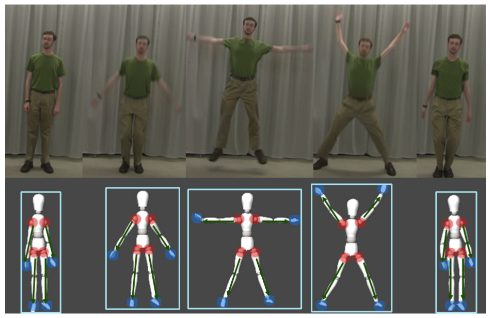
**Figure 5:** *Use of Kinect for query specification. A set of user-specified motion keys are computed from the Kinect motion data which is used for searching the database to return motion clips that are similar to the recorded motion.*

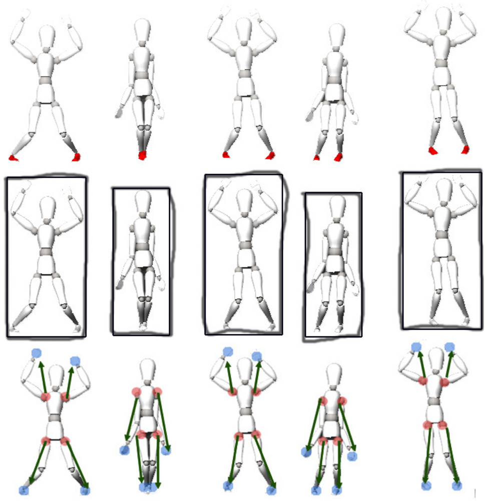
**Figure 6:** *Query specification for a jumping jack motion: A support key, bounding box key (specified using sketches) and, orientation keys for the arms and legs are used to specify the input query.*

**Locomotion.** End users can very quickly create simple queries to define motions such as walk cycles, jumping, crouching, and different arm gestures. A walk cycle can be defined by specifying a left foot plant followed by a right foot plant, with no flight phase. Introducing a flight phase in between returns running motions. By placing additional constraints on the displacement of the center of mass $K^{\text{com-disp}}_{\text{s}}(u)$ and the relative orientation of the swing foot $K^{\text{orient}}_{\text{foot}}(u)$, we can return motions with long strides, as well as different turning and side-stepping motions. Different jumping variations (e.g., forward jump, jump in place, hopping on one leg) can be retrieved by specifying the COM displacement and using different support configurations. Effort motion keys are very useful for characterizing the dynamics of motion. Motions where the character is moving along a straight line are characterized by direct space effort $K^{\text{space}}_{\text{body}}(u)$. For indirect effort, results are returned where the character is walking in circles, with little or no net displacement. Stiff walking motions have sudden time effort due to acceleration peaks

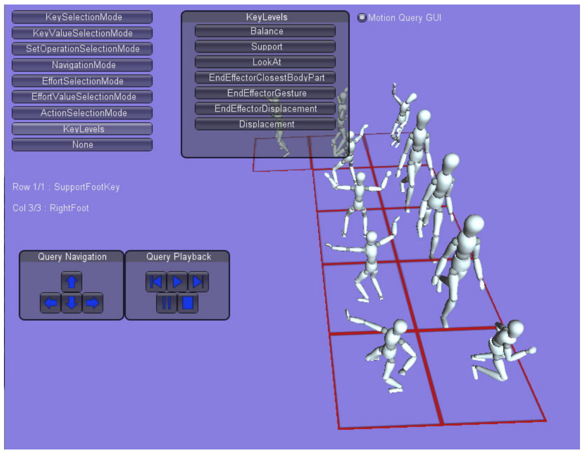

**Figure 7:** *Graphical User Interface for Motion Querying and Retrieval.*

at the end of each walk cycle. A fluid walking motion, in contrast has sustained time effort.

**Arm Gestures.** Arm gestures are queried by checking if the displacement $K_{\mathrm{s}}^{\mathrm{disp}}(u_{\mathrm{k}})$ of the hands crosses a maximum threshold. Results can be easily filtered to be specific to either arm, or by using $K_{\mathrm{s}}^{\mathrm{orient}}(u_{\mathrm{k}})$ to detect arm gestures with a specific orientation. Furthermore, the effort keys are particularly useful to query for gestures with specific dynamic properties. Strong gestures such as a punch, or light gestures such as slow reaching motions can be queried using $K_{\mathrm{s}}^{\mathrm{weight}}(u_{\mathrm{k}})$. $K_{\mathrm{s}}^{\mathrm{space}}(u_{\mathrm{k}})$ is used to specify constraints on the spatial pattern of the motion. Please refer the accompanying video and supplementary document for different combinations of keys used to retrieve a variety of arm gestures.

**Climbing.** We detect climbing motion by its center of mass displacement $K^{\text{com-disp}}(u)$ along the vertical axis. Specifying a sequence of gradually increasing vertical displacement values, we can search for motions like climbing up a ladder. Introducing a displacement along the horizontal axis returns results where the center of mass takes a more diagonal trajectory, such as climbing a flight of stairs.

**Jumping Jacks.** Exercise motions like jumping jacks require full-body coordination with the arms and legs following a sequence of gestures during the motion. Figure 6 illustrates the specification of arm and leg orientation, the support key, as well as the bounding volume of the body to describe a jumping jack motion. A real benefit of our approach is to incrementally add more constraints using different motion keys to filter the search. For example, we can specify that that the two hands must make contact in the middle of the motion, or change the desired orientation of the hands and legs to return more specific results.

**Cartwheel.** A character performing a cartwheel motion can be defined using the following regular expression of $K^{\mathrm{support}}(u)$ values:

$$\begin{aligned} \mathrm{q}(K^{\mathrm{support}}(u)) \quad \leftarrow \quad & (\mathtt{BF}^{+} \cdot \mathtt{RF}^{+} \cdot \mathtt{BH}^{+} \cdot \mathtt{LF}^{+} \cdot \mathtt{RF}^{+} \cdot \mathtt{BF}^{+}) \\ & |(\mathtt{BF}^{+} \cdot \mathtt{LF}^{+} \cdot \mathtt{BH}^{+} \cdot \mathtt{RF}^{+} \cdot \mathtt{LF}^{+} \cdot \mathtt{BF}^{+}) \end{aligned} \tag{20}$$

where `B`,`F` and, `H` stand for `Both`, `Foot` and, `Hand` respectively. Note that the regular expression must also specify the presence of zero or more `Blank` frames after each support constraint, which has been omitted from Equation 20 for ease of exposition. A flight phase can also be introduced into the cartwheel by specifying that no hands must touch the ground in the middle of the motion. Other filters may also be applied by using combinations of other motion keys. Figure 1 illustrates the specification of support and shape keys for retrieving a cart wheel motion.

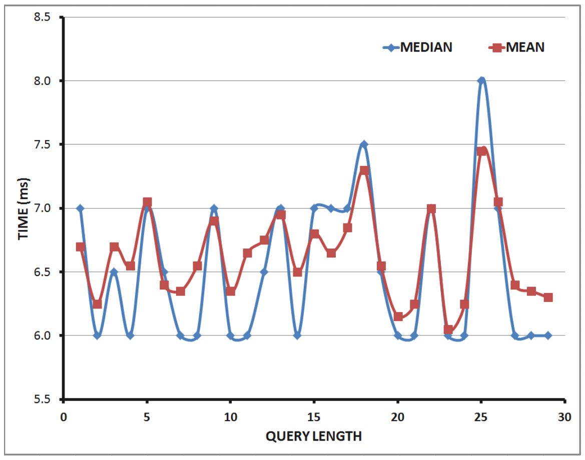

**Figure 8:** *Search times for a single trie with increase in query length. The trie is populated with computed key values for all* 6121 *motions in the CMU and HMD motion databases.*

**Alternate User Interfaces.** Figure 7 illustrates a prototype graphical user interface for querying and motion retrieval. Motion keys can also be mapped to more intuitive forms of specification. For example, a user can sketch a sequence of rectangles to represent the change in shape of a character over the course of a motion, as illustrated in Figure 6. We also demonstrate the use of the Microsoft Kinect sensor as an alternate form of query specification. A user records a query motion using the Kinect Sensor which is converted to motion data using the OpenNI API and is used to compute a set of user-specified motion keys for the input motion. Note that the algorithm used is the same for both offline key computation and Kinect input analysis, assuring compatible key-frame selection semantics. The sequence of key values serves as the input query which is used for retrieving similar motions. Figure 5 illustrates the use of the Kinect to search for a jumping-jack motion in the database. Here, the user specified the use of arm gestures and the shape of the body as the keys used to generate the query.

**Performance.** A big advantage of using tries for indexing is that the size of the database does not affect the the time complexity of the search, which is a constant time operation. The search time depends on the number of keys used in the query, where each key maps to an independent search on a trie. Figure 8 illustrates performing searches on a single trie with increase in query length. We observe that the average time to perform a single trie search was 6.5 ms. Also, specifying temporal constraints requires a filtering process that is linear in the number of results returned. During our experiments, we observed that the average time to search was $(25-40$ ms) ms when $4-6$ keys were used in combination.

## 8 Conclusion

The motion keys described in the paper are *not* a comprehensive set of all motion features, but are intended to provide a medium for researchers and end-users to define different categories of motion features for use in their applications. One category of LMA that is still under active exploration is the theories of *Space* which describe motion in connection with the environment and spatial patterns of movement. For future work, we would like to incorporate these elements into our set of motion keys to facilitate description of character interaction with its environment. An open question that we, and the community at large strives to answer is the ability to define *coverage* for a set of motion features, and automatically derive the minimal set of features that can completely characterize the extremely high dimensional space of human motion.

For indexing into the database, we use a trie-based data structure that supports efficient sub-sequence matching and fuzzy searches. A key advantage of tries is that the search times are independent of the size of the motion database with a worst case time complexity of $O(l)$, where $l$ is the length of the longest motion. Tries are also highly cache local and extremely parallelizable. For future work, we will explore the use of recent advancements in graphics hardware to further accelerate search times. Also, static tries with no more insertions and deletions can also be greatly compressed by merging common branches, which would greatly reduce the memory footprint.

Querying motions using regular expressions provide a flexible interface for making general as well as specific queries. However, specifying regular expressions can be cumbersome and require the user to have prior knowledge of the different motion keys and an interpretation of their values. The current system is not immediately accessible to end users for whom it may be prohibitive to specify complex regular expressions. The use of sketches and other alternate user interfaces such as the Kinect are promising directions to provide a more intuitive means of query specification, but may not be applicable for all the keys described in this paper. Based on our experience, we envision the development of specific interfaces designed to support each key category.

Our method returns motion subsequences which satisfy conditions on the numerical values of one or more keys, as specified in the query. Motions that are not useful to the user may be pruned by adding additional filters to the query. However, it is possible that useful motions may also be pruned as a result, producing false negatives. This is due to the unintuitive mapping of key values to body configurations, and the discretization of key values. For future work, we would like to carry out human factors experiments to validate the results of our approach, and determine the sensitivity of the results on the coarseness of key discretization.

The provision of so many different motion keys (43 in our current implementation) can provide flexibility to the user but can also become burdensome as the user is required to choose which key(s) to use. This is evident when using the Kinect sensor to input a query motion as the user had to manually specify which keys would be most applicable. Automatic feature selection and generating a minimal, yet complete feature set is an open area of research.

## References

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. SIGGRAPH, 402–408.

ASSA, J., CASPI, Y., AND COHEN-OR, D. Action synopsis: pose selection and illustration. SIGGRAPH '05, 667–676.

BARBIČ, J., SAFONOVA, A., PAN, J.-Y., FALOUTSOS, C., HODGINS, J. K., AND POLLARD, N. S. Segmenting motion capture data into distinct behaviors. In *GI '04*, 185–194.

BARTENIEFF, D. I. 1972. *Effort-shape analysis of movement: The unity of expression and functions*. Arno Press Inc., New York.

BOUCHARD, D., AND BADLER, N. 2007. Semantic segmentation of motion capture using laban movement analysis. IVA '07, 37–44.

BOUVIER-ZAPPA, S., OSTROMOUKHOV, V., AND POULIN, P. 2007. Motion cues for illustration of skeletal motion capture data. NPAR, 133–140.

CHAI, J., AND HODGINS, J. K. 2005. Performance animation from low-dimensional control signals. SIGGRAPH '05, 686–696.

CHAO, M.-W., LIN, C.-H., ASSA, J., AND LEE, T.-Y. 2011. Human motion retrieval from hand-drawn sketch. *IEEE Transactions on Visualization and Computer Graphics 99*.

CHI, D., COSTA, M., ZHAO, L., AND BADLER, N. The EMOTE model for effort and shape. SIGGRAPH '00, 173–182.

CHOI, M. G., YANG, K., IGARASHI, T., MITANI, J., AND LEE, J. 2012. Retrieval and visualization of human motion data via stick figures. *CGF 31*, 7pt1, 2057–2065.

CMU. 2003. Carnegie mellon univ. mocap database (NSF grant 0196217) http://mocap.cs.cmu.edu.

DENG, Z., GU, Q., AND LI, Q. 2009. Perceptually consistent example-based human motion retrieval. I3D '09, 191–198.

FALOUTSOS, C., RANGANATHAN, M., AND MANOLOPOULOS, Y. 1994. Fast subsequence matching in time-series databases. ACM, New York, NY, USA, SIGMOD '94, 419–429.

FAUVET, B., AND BOUTHEMY, P. 2004. A geometrical key-frame selection method exploiting dominant motion estimation in video. In *IEEE International Conference on Content-based Image and Video Retrieval*, 419–427.

FENG, T.-C., GUNAWARDANE, P., DAVIS, J., AND JIANG, B. 2008. Motion capture data retrieval using an artist's doll. In *International Conference on Pattern Recognition, 2008.*, 1–4.

FREDKIN, E. 1960. Trie memory. *Commun. ACM 3* (September), 490–499.

GLEICHER, M. 1998. Retargetting motion to new characters. SIGGRAPH, 33–42.

KEOGH, E., PALPANAS, T., ZORDAN, V. B., GUNOPULOS, D., AND CARDLE, M. 2004. Indexing large human-motion databases. VLDB, 780–791.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph. 23* (Aug.), 559–568.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Trans. Graph. 21* (July), 473–482.

KRÜGER, B., TAUTGES, J., WEBER, A., AND ZINKE, A. 2010. Fast local and global similarity searches in large motion capture databases. SCA, 1–10.

LABAN, R. 1971. *The Mastery of Movement*. Plays, Inc.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM TOG 21*, 491–500.

LI, Q. L., GENG, W. D., YU, T., SHEN, X. J., LAU, N., AND YU, G. 2006. MotionMaster: authoring and choreographing kung-fu motions by sketch drawings. SCA, 233–241.

LIU, G., ZHANG, J., WANG, W., AND MCMILLAN, L. 2005. A system for analyzing and indexing human-motion databases. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, ACM, New York, NY, USA, SIGMOD '05, 924–926.

MALETIC, V. 1987. *Body, Space, Expression: The Development of Rudolf Laban's Movement and Dance Concepts*. Mouton de Gruyte, New York.

MIXAMO, 2010. Mixamo: Animation in seconds http://www.mixamo.com/.

MÜLLER, M., AND RÖDER, T. 2006. Motion templates for automatic classification and retrieval of motion capture data. SCA, 137–146.

MÜLLER, M., RÖDER, T., AND CLAUSEN, M. 2005. Efficient content-based retrieval of motion capture data. *ACM Trans. Graph. 24* (July), 677–685.

NUMAGUCHI, N., NAKAZAWA, A., SHIRATORI, T., AND HODGINS, J. K. 2011. A puppet interface for retrieval of motion capture data. SCA, 157–166.

OKAJIMA, S., WAKAYAMA, Y., AND OKADA, Y. 2012. Human motion retrieval system based on LMA features using interactive evolutionary computation method. In *Innovations in Intelligent Machines 2*, vol. 376. 117–130.

REN, L. 2006. *Statistical Analysis of Natural Human Motion for Animation*. PhD thesis, Pittsburgh, PA, USA.

SAKAMOTO, Y., KURIYAMA, S., AND KANEKO, T. Motion map: image-based retrieval and segmentation of motion data. SCA '04, 259–266.

TAUTGES, J., ZINKE, A., KRÜGER, B., BAUMANN, J., WEBER, A., HELTEN, T., MÜLLER, M., SEIDEL, H.-P., AND EBERHARDT, B. 2011. Motion reconstruction using sparse accelerometer data. *ACM Trans. Graph. 30*, 3 (May), 18:1–18:12.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: an interface for sketching character motion. *ACM Trans. Graph. 23* (Aug.), 424–431.

TORRESANI, L., HACKNEY, P., AND BREGLER, C. 2006. Learning motion style synthesis from perceptual observations. In *NIPS*, 1393–1400.

UTA, 2011. Human motion database: University of Texas at Arlington http://smile.uta.edu/hmd/.

WAKAYAMA, Y., OKAJIMA, S., TAKANO, S., AND OKADA, Y. 2010. IEC-based motion retrieval system using Laban Movement Analysis. In *KES*, 251–260.

WU, S., WANG, Z., AND XIA, S. 2009. Indexing and retrieval of human motion data by a hierarchical tree. VRST, 207–214.

XIAO, J., ZHUANG, Y., YANG, T., AND WU, F. 2006. An Efficient Keyframe Extraction from Motion Capture Data. *Advances in Computer Graphics*, 494–501.

YU, T., SHEN, X., LI, Q., AND GENG, W. 2005. Motion retrieval based on movement notation language. *CAVW 16*, 273–282.

ZHAO, L., AND BADLER, N. I. 2005. Acquiring and validating motion qualities from live limb gestures. *Graph. Models 67*, 1–16.