Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

Motion-Motif Graphs

Philippe Beaudoin Stelian Coros Michiel van de Panne Pierre Poulin

University of British Columbia Université de Montréal

Abstract

We present a technique to automatically distill a motion-motif graph from an arbitrary collection of motion capture data. Motion motifs represent clusters of similar motions and together with their encompassing motion graph they lend understandable structure to the contents and connectivity of large motion datasets. They can be used in support of motion compression, the removal of redundant motions, and the creation of blend spaces. This paper develops a string-based motif-finding algorithm which allows for a user-controlled compromise between motif length and the number of motions in a motif. It allows for time warps within motifs and assigns the majority of the input data to relevant motifs. Results are demonstrated for large datasets (more than 100,000 frames) with computation times of tens of minutes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Animation

1. Introduction

Data-driven character animation requires structured motion capture data in order to succeed. Motion graphs allow original sequences of motion data to be resequenced by identifying similar pairs of frames and are therefore one straightforward way to provide structure. Deeper analysis of motion data can be used to build more abstract models of how a motion can evolve over time, such as hidden Markov models (HMMs), switched linear dynamical systems (SLDSs), and Gaussian process dynamical models (GPDMs). However, these techniques still have significant limitations when it comes to motion synthesis. The development of *parametric* [HG07] or *fat* [SO06] motion graphs, provides a more recently developed middle ground in terms of abstraction. These techniques develop motion graphs that model the connectivity of *groups* of similar motions as opposed to individual motions sequences. In comparison to statistical models, these models are more transparent to end users because they are more consistent with current methodologies for games and interactive simulations. Their resequence-and-blend based approach for generating motions is also well known to produce high quality results.

We propose algorithms for automatically identifying groups of similar motion subsequences, which we call *motion motifs*, and building a motion graph on top of these motifs. We call the final structure a motion-motif graph in deference to the common use of the word *motif* in the context of data mining for describing approximately repeating patterns.

A number of desired features of motion-motif graph algorithms are as follows: (a) The algorithms should be automatic, efficient, and scalable to large datasets; (b) Extracted structure should be stable in the face of added noise; (c) Motifs should be meaningful in terms of their semantics, their perceptual nature, or their functionality when used for motion synthesis; (d) Similar motions of different durations should still be able to exist within the same motif; (e) Motifs themselves should be able to exist across different time scales if this is supported by the data; (f) Most of the motion data should be incorporated into motifs, i.e., the problem is not just to extract a few of the most identifiable motifs; and (g) There should be some control over the desired type of partitioning, i.e., motifs should not be biased exclusively towards choosing either long motifs or common motifs. The algorithms developed in this paper strives to meet these criteria.

The paper is organized as follows. Section 2 discusses related work. Section 3 details the algorithm for motif discovery. Section 4 shows how the motion motifs can be used to produce a graph structure. Section 5 presents and analyzes experimental results for motion motif extraction and their use for motion synthesis. Section 6 provides concluding remarks.

2. Related Work

The past decade has spawned a growing body of work that examines the analysis, synthesis, compression, and retrieval of human motion data. We structure our survey of related work according to the primary goal of the proposed techniques, but note that there are frequently deep and sometimes undocumented connections between the various techniques.

Automatic **segmentation** of motions into distinctive smaller fragments has been investigated in support of a number of applications, including motion compression [LM06], motion classification [BSP*04, SB05, FMJ02], and motion retrieval [LZWM05, SB05]. Approaches used for segmentation include angular-velocity zero-crossing detection [FMJ02], entropy metrics [SB05], and temporally local models based on probabilistic principal component analysis (PCA) [BSP*04, LM06].

Motion query techniques extract structure in order to be able to efficiently identify motions that are similar to a query motion. This has recently been an active area of research. One common approach is to develop and apply a segmentation rule and then cluster the resulting fragments [BSC05, LZWM05]. Motion queries are then performed by looking for a given cluster transition signature. An alternate model is to look for patterns in binary-valued relational features [MRC05, MR06] or extracted keyframes [SB05], and to construct efficient searches based on these. Search strategies can also be informed by user-weighted notions of important features [FF05] and can be made to efficiently support time warps [KPZ*04, SB06]. Another approach builds a precomputed 'match web' from a pairwise comparison of all frames in the motion corpus, which can then be used to quickly retrieve motions that are similar to query motions also selected from the corpus [KG04].

Statistical models of motion are intended to be general and can in theory be used for both analysis and synthesis. In parametric statistical models, the original motion data is discarded and thus original motion retrieval is not an option. Hidden Markov models [BH00, TH00] and switched linear dynamical systems [PRM00, LWS02] have been applied to motion synthesis. Stochastic models have also been used to construct natural-motion classification oracles [RPE*05].

Motion graphs aim to address the motion resequencing problem by automatically identifying points where motions are sufficiently similar that they support transitions between motions, and hence allow resequencing. They have been introduced in various forms in recent years [TH00, AF02, KGP02, LCR*02, LWS02] and resemble the *move trees* that have long been a staple for game-based character motion.

***Fat* or *parametric* motion graphs** [SO06, HG07] support both motion resequencing and motion blending. This is achieved by building a motion graph from sets of parameterized motions, where motions within a set can be blended. Sets of parameterized motions can be constructed manually [PSKS04] or with the help of a motion query algorithm that can retrieve similar motions given an example query motion [KG04, KS05]. The parametric motions in [HG07] are constructed using the techniques developed in [KG04] to efficiently find and resample sets of motions that are similar to a given query motion. A user-specified distance threshold value determines how large the returned set of similar motions will be. Where the similar motions are considered to start and end is implicit in the start and end of the query motion chosen by the user. As a result, the nature and duration of each parameterized motion set is in effect user-specified. The parametric motions in [KS05] are identified with the help of string matching, where the alphabet denotes the various support phases (right leg, left leg, double support, flight) for walking and running.

Motif discovery techniques have the general goal of extracting approximately repeating patterns in data, including sequential data such as time series. Finding motifs in univariate sequential data has applications to finding common structural patterns in biological applications [BE94, CKL03] and has the longest history, although advances continue to be made, e.g., [LKLC03]. This last technique has been used to discover motifs from low-dimensional representations of motion capture data [AAT*06]. Extending motif discovery techniques to real-valued multivariate time series is currently a highly active topic of research [Oat02, TIU05, YKM*07, MSEI07, MIES07b, MIES07a]. Recent work applies motif mining to motion capture data [MYHW08]. It produces a fixed partitioning that favors long motifs. It does not model connectivity between motion motifs and it mentions building motion graphs around motifs for complex motion data sets as future work.

Our work is inspired by many of the recent developments in motif discovery algorithms, and seeks to develop algorithms that meet the desired features listed in the introduction. Current motif-discovery algorithms often forego (d), (f), and (g) from the feature list and provide a limited evaluation with respect to (b), (c), and (f). To the best of our knowledge, the algorithms and results presented in this paper are also novel with respect to: (1) allowing for and demonstrating partitioning control using a single parameter to specify the desired length-vs-cardinality tradeoff that naturally exists for motifs in many datasets; (2) demonstrating the effectiveness of using vector quantization to represent the multivariate motif discovery problem in terms of strings; (3) demonstrating results for motion capture datasets having more than 100,000 frames of data; (4) development and illustration of the embedding motion-motif graphs and related demonstrations of motion synthesis using this graph; (5) the use of coverage percentage and average frame error to evaluate motion motif graphs; (6) demonstrating the ability to exploit the coherency of motions in motion motifs for the purposes of compression.

A number of evaluation metrics have been suggested for

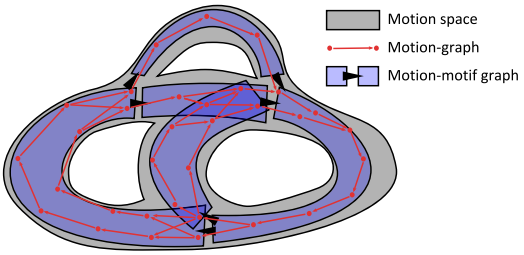

Figure 1: *Motion live in a constrained and continuous space, represented here in an abstract form by the gray surface. Motion graphs approximate this space using a number of discrete paths that can branch into one another. Motion-motif graphs represent it using a patchwork of continuous subregions.*

motion motifs in the context of motion analysis for activity classification [MWS*06]. Our evaluations aim to demonstrate that the desired features listed in the introduction are satisfied in the context of motion synthesis.

The final result of our algorithm is a motion-motif graph, which reveals a more abstract, higher-level structure as compared to motion graphs. This is illustrated in Figure 1.

3. Motion Motif Extraction

3.1. Overview

We begin our description of the system, shown in block-diagram form in Figure 2, by explaining a toy example. Figure 3 shows how motion motifs are extracted for a set of motions in a simple two dimensional space. The goal of the algorithm is to produce motion-motif partitions. As shown in Figure 3 (f) and (g), there exists more than one way to naturally partition the motions into motion motifs. A parameter $\rho \in [0.2, 5]$ controls the preferred type of partitioning; $\rho > 1$ gives preference to longer motion motifs, at the possible expense of having fewer motions in any given motif.

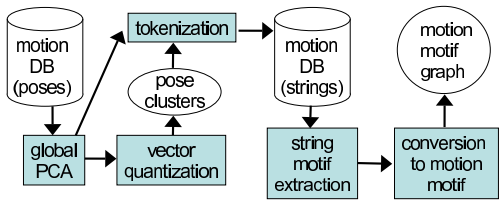

Figure 2: *System Overview.*

The first step in the process is to reduce the dimensionality of all poses through the application of PCA. This is followed by vector quantization, illustrated in Figure 3(a), for all poses. Vector quantization is performed using K-means clustering. Individual poses are assigned labels, here shown as letters, according to their assigned cluster. The motions can now be converted into motion strings using the letters associated with each pose. Sequential repetitions of letters are removed. All the motion sequences in the motion corpus can be represented as strings and then further concatenated into a single long string representing all motions. Figure 3(b) shows a portion of this long string for the toy example. The collection of motions that runs horizontally is represented by the substring GOACEN, and thus multiple occurrences of this substring can be observed. An additional *partition flag*, $p(j)$, is used to flag points where adjacent motion sequences have been concatenated. Later, $p(j)$ will also be used to flag subsequences that have already been incorporated into motion motifs and that are therefore ineligible for use in new motion motifs.

The construction of motion motifs requires knowing which letters are 'nearby' other letters. This can be captured using an *adjacency matrix* **A**, as illustrated in Figure 3(c). This will be used to help identify non-identical substrings that nevertheless represent similar motions. For example, the motions represented by GOACEN and GOACHCEN are very similar and should be eligible to be clustered together.

The clustering algorithm works by randomly generating a number of seed substrings ranging in size from a minimum to a maximum length. For each of these seed substrings, a fast string matching algorithm is applied in order to build a temporary string motif. A particular instance of the substring HCEN is chosen in our example, as shown in Figure 3(d). The complete details of this process will be described in shortly (§3.2). Once the largest string motif is found, all the substrings that it encompasses are marked as 'partitioned', as shown in Figure 3(e), and the process is then repeated with a new set of seed substrings.

3.2. Pose Preprocessing

In this section, we give a more detailed description of the pose preprocessing and introduce the notation that is used when describing subsequent steps of the algorithm. Each pose in the dataset is expressed using a vector containing the joint angles, the root height, the root pitch and roll angles, the root horizontal speed, and the root angular speed with respect to the vertical Y axis. This makes the pose representation independent of the facing direction and the location on the plane, as is commonly desired. The dataset can be expressed as an ordered list of n vectors $\Theta(i)$, $1 \leq i \leq n$. A submotion $\Theta(i), \Theta(i+1), ..., \Theta(i')$ is denoted by a pair of indices (i, i').

All test sequences we use to build the dataset share the same bone hierarchy, leading to a 62-dimensional pose space. We perform a global PCA where all $\Theta(i)$ are projected into a subspace of fewer dimensions to yield $\hat{\Theta}(i)$. We perform projections that keep 97% of the total variance,

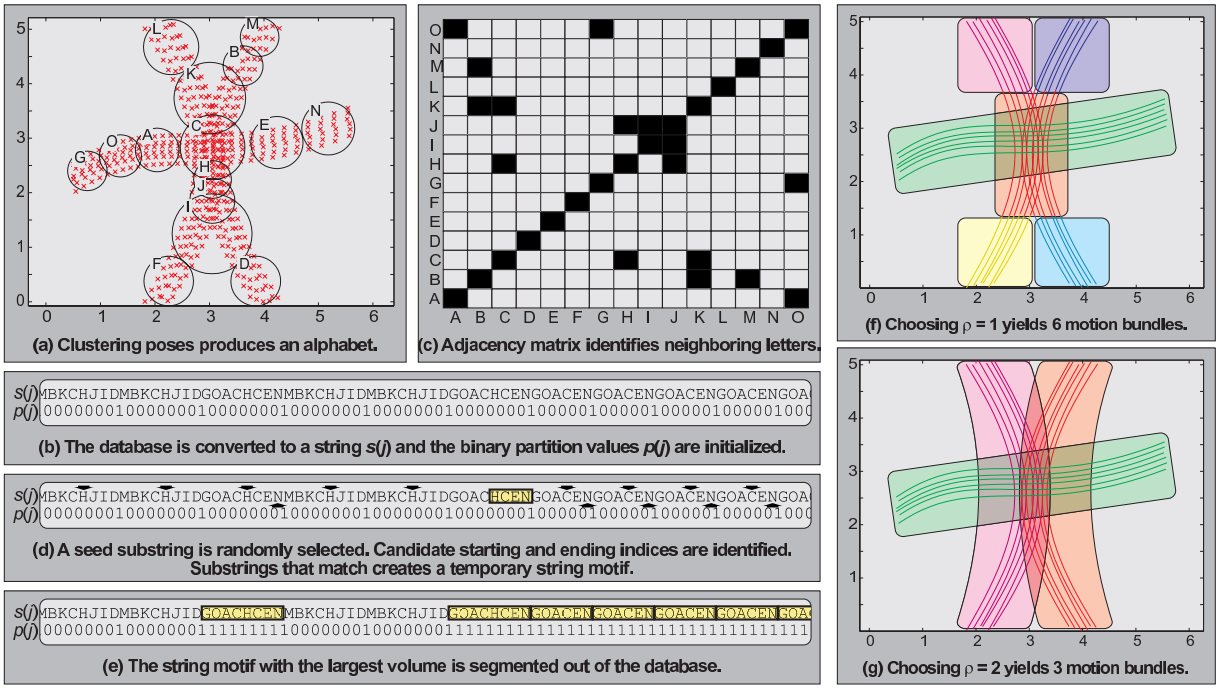

Figure 3: *The various steps of the motion-motif algorithm on a 2D toy example. The frames for this toy example are marked in red in (a). The motion sequences from which they come are the curves traced in (f) and (g).*

corresponding to keeping 12–18 principal components, depending on the dataset. Global PCA helps speed up the pose clustering process.

Vector quantization is performed using a fast K-means clustering algorithm. The total number κ of clusters needs to be specified by the user and the impact of this parameter is explored in detail later (§5.2). Each cluster can be assigned a letter from a κ-letter alphabet. All motions can therefore be represented as a n-letter string by associating each pose $\hat{\Theta}(i)$ with the nearest cluster. This string can be further simplified by removing consecutive repetitions of the same letter, leading to a m-letter string $s(j)$ with $1 \leq j \leq m$. Moreover, for each letter $s(j)$ we can store the set of consecutive frames $\mathcal{F}(j)$ that map to this letter. The mean of this set is noted $\bar{\mathcal{F}}(j)$. A substring $s(j), s(j+1), ..., s(j')$ is denoted by a pair (j, j').

The quantization process makes it possible for two poses close to one another to be assigned to different clusters. To overcome this problem, we build an *adjacency matrix* $\mathbf{A}$ that lets us identify neighboring clusters. To build this matrix we first attach an hypersphere to each cluster center. The radius of an hypersphere is computed using the distance of the furthest data point that was assigned to it. Two clusters are then deemed adjacent if their corresponding hyperspheres intersect.

3.3. String Motif Discovery

The motion-motif creation process works by first building motifs from the string-based representation of motions, followed by a second step that maps the result back to the original motions. We postpone the discussion of this second step until Section 3.5. String-motif creation works by iteratively partitioning the motion string. To track the algorithm as it progresses, a binary partition value is associated with each transition between two consecutive letters of the string. This value is noted $p(j)$ with $1 \leq j \leq m-1$ so that $p(j)$ corresponds to the interval between $s(j)$ and $s(j+1)$. By default, all values of $p(j)$ are initialized to 0, indicating that the whole string is unpartitioned. It is possible to initialize some values of $p(j)$ to 1 to indicate that two consecutive letters belong to different motion sequences. This way, the algorithm will never include that interval into a string motif.

The values of $p(j)$ separate the string into partitioned letters and unpartitioned letters. We say that letter $s(j)$ is *partitioned* if and only if $p(j-1) = p(j) = 1$, and that it is otherwise *unpartitioned*. In the same way, substring (j, j') is partitioned if and only if $p(j) = p(j+1) = ... = p(j'-1) = 1$, and unpartitioned if and only if $p(j) = p(j+1) = ... = p(j'-1) = 0$. It is possible for a substring to be neither partitioned nor unpartitioned.

The creation of every string motif first requires us to generate a random set of seed substrings. The selected substrings must be unpartitioned and their length must fall

within the user-defined range $[\lambda^-, \lambda^+]$. In practice, we generate this set by randomly selecting frequently occuring letters and producing all unpartitioned substrings that contain these letters and whose length fall in the given range.

For each of these seed substrings, we look through the full motion string for all similar and unpartitioned substrings. Identifying good motion motifs requires a definition of 'similar' that is broader than simply identifying copies of the substring in question. Details of this process are given in Section 3.4. The end product of the overall search is a distinct set of matching substrings for each possible choice of seed substring.

Each of these sets of substrings can be regarded as defining a different potential motion motif, and we thus need to choose which one will give the largest motion motif. We define a *motif volume* V related to the motif dimensions as follows:

$$V = (\mathcal{B}_h - 1)(\mathcal{B}_w)^\rho \tag{1}$$

where $\mathcal{B}_h$ is the motif height (the number of substrings in the motif) and $\mathcal{B}_w$ is the motif width (the average number of frames in the motions corresponding to the included substrings). ρ is the parameter that is used to influence the preferred lengths of motifs.

The string motif having the largest volume V is identified and kept. If all potential motifs contain a single substring, then a two-letter substring in randomly selected from the seed substrings and segmented out, making sure the algorithm eventually terminates. Following that, the values of $p(j)$ are updated so that all substrings in the chosen motif are now marked as being partitioned. The algorithm then repeats using a newly chosen seed point.

3.4. Identifying Similar Substrings

The process of identifying substrings similar to a seed substring is inspired by the search algorithm proposed by [KG04], although we compare pose clusters rather than poses. For a given substring, we need to identify all other substrings with which we can build a cluster-to-cluster registration curve that satisfies some constraints. We begin by defining the binary relation $M(ss_1, ss_2)$ that is true if and only if substring ss_1 matches substring ss_2.

Suppose we have two substrings ss_1 and ss_2 identified respectively by (j_1, j_1') and (j_2, j_2'). We can build a substring-to-substring binary accessibility matrix **SSA** as

$$\mathbf{SSA}_{[a-j_1, b-j_2]} = \mathbf{A}_{[s(a), s(b)]} \tag{2}$$

with $j_1 \leq a \leq j_1'$ and $j_2 \leq b \leq j_2'$. For $M(ss_1, ss_2)$ to be true there must exist, within **SSA**, a valid path of 1's starting from cell $\mathbf{SSA}_{[0,0]}$ and ending at $\mathbf{SSA}_{[j_1'-j_1, j_2'-j_2]}$. Suppose we have such a candidate path going through cells $[a_1, b_1], [a_2, b_2], ..., [a_l, b_l]$, with $a_1 = b_1 = 0, a_l = j_1' - j_1, b_l = j_2' - j_2$, and where $\mathbf{SSA}_{[a_i, b_i]} = 1$ for all i. This path is valid if and only if $a_i \leq a_{i+1} \leq a_i + 1$ and $b_i \leq b_{i+1} \leq b_i + 1$ for $0 \leq i < l$.

The existence of a path through matrix **SSA** implies the existence of a valid time alignment curve between the motions corresponding to ss_1 and ss_2, up to the granularity induced by the vector quantization. To make sure this time-alignment does not introduce too much distortion, we require the user to provide a slope limit γ and use the criterion introduced by Kovar and Gleicher [KG04]. To better evaluate the slope of the time alignment curve, we use the frames defined by $\bar{\mathcal{F}}$ rather than simple indices into the string.

Provided we have a seed substring $ss_0 = (j_0, j_0')$, we can build the set of all substrings for which $M(ss_0, ss_a)$ holds true. To do this efficiently, we first identify a short list of substrings over which the relationship should be tested. We do this by identifying letters that are good start and end candidates. A letter is a start candidate (respectively, an end candidate) if it matches $s(j_0)$ (respectively $s(j_0')$) and if its cluster is closer to $s(j_0)$ (respectively $s(j_0')$) than the clusters of the two letters directly beside it. Iterating over the start and end candidates let us quickly generate a limited number of candidate subtrings to use for matching.

3.5. From String Motifs to Motion Motifs

String motifs need to be mapped back to particular frame sequences of the original motions. While this step is largely trivial, there remains a small-but-non-negligible issue. Multiple successive frames of a motion sequence will often map to the same letter in a substring. Arbitrarily choosing one of these frames for the first or last letter of a substring may lead to motions that are not as well aligned as they could be. It is thus advantageous to further optimize, at the frame level, where a given motion enters and exits a motion motif.

To do so, we first identify the indices of all letters of the dataset appearing at the beginning or the end of a substring. The set containing these indices is noted $\mathcal{J}$. Mapping back from string to motion is then simply a matter of finding a frame $f(j)$ for each $j \in \mathcal{J}$.

Given such a mapping, we can evaluate the quality of the motion alignment within a motif. To do so we define the alignment score for motion motif $\mathcal{B}$ as

$$\sum_i |\Theta(i) - \bar{\Theta}^-| + \sum_{i'} |\Theta(i') - \bar{\Theta}^+| \tag{3}$$

where i and i' respectively indicate all the starting frames and all the ending frames of motions in $\mathcal{B}$, while $\bar{\Theta}^-$ and $\bar{\Theta}^+$ respectively refer to the average starting and ending poses. The total alignment score for a given assignment of the $f(j)$ is the sum of the alignment scores over all the motion motifs.

We look for an assignment that minimizes the total alignment score. To do so, we rely on the following stochastic search algorithm. First, we use the initial guess $f(j) = \bar{\mathcal{F}}(j)$ for all $j \in \mathcal{J}$. Then an index $j' \in \mathcal{J}$ is randomly selected

and a direct search is performed within $\mathcal{F}(j')$ to find the assignment $f(j')$ that minimizes the total alignment score. The algorithm is repeated with a newly selected random index until the total alignment score cannot be improved or for a predefined number of iterations.

The final step of the conversion to motion motifs consists of computing a precise time-alignment for the motions within a motif. We do this by first identifying, within the motif, the motion closest to the average. Each other motion is then time-aligned with this one through a registration curve computed using dynamic time-warping.

4. Embedding Motion Motifs into Graphs

Motion motifs can be seen as a way of folding the motion dataset so that different submotions overlap. This naturally creates a directed graph structure where each node is a motion motif and each edge is a valid transition from one motion motif to another one. For convenience, we call this a motion-motif graph. Parameterized motion graphs [HG07] use a similar graph structure. It has some advantages over the motions-as-edges, nodes-as-transitions structure used in traditional motion graphs [KGP02] and fat graphs [SO06]. Specifically, in a nodes-as-transitions motion graph, any motion coming into a node must be able to transition into any motion leaving from that node. This makes it difficult to create good hub nodes with rich connections and also necessitates short-duration blends from incoming onto outgoing motions. In contrast, the nodes-as-motions model supports both long blend intervals during transitions, as well as rich parameterizations of motions using multi-way blends.

Building a motion-motif graph from the motion-motif structure is straightforward. First, a node B is created for each motif $\mathcal{B}$. Then, an edge $B_1 \rightarrow B_2$ is added if there exist poses i_1, i_2, i_3, i_4 such that sequence $(i_1, i_2) \in \mathcal{B}_1$, sequence $(i_3, i_4) \in \mathcal{B}_2$ and either $i_2 = i_3$ or sequence (i_2, i_3) is not part of any motif. In our graphical representation, nodes are represented as boxes with an height proportional to $\mathcal{B}_h$ and a width proportional to $\mathcal{B}_w$. Graph layout is performed automatically using the publicly available GraphViz tool.

It is possible to produce an animation that follows any valid path through this graph. To do so, we exploit the similarity of motions within a motif and blend from the entry motion to the exit motion. More precisely, assume a desired path $B_1 \rightarrow B_2 \rightarrow B_3$. Then $B_1 \rightarrow B_2$ guarantees that we can find i_1, i_2, i_3, i_4 such that $(i_1, i_2) \in \mathcal{B}_1$ and $(i_3, i_4) \in \mathcal{B}_2$. Similarly, $B_2 \rightarrow B_3$ guarantees that we can find i'_3, i'_4, i'_5, i'_6 such that $(i'_3, i'_4) \in \mathcal{B}_2$ and $(i'_5, i'_6) \in \mathcal{B}_3$. To play the motion corresponding to B_2, we simply blend smoothly from submotion (i_3, i_4) to submotion (i'_3, i'_4).

Even though the blended animations are similar and well-aligned, it is possible that their contact constraints differ. Various strategies could be used to solve this problem, such as inverse-kinematics-based corrections. Such techniques are already well studied and thus we do not address them further in this paper. The animations presented in the accompanying video do not use such strategies and moderate foot-skate and other minor motion artifacts can sometimes be observed.

5. Results

5.1. Extracting Motion Motifs

We tested the motion-motif algorithm on various collections of motions, ranging in size up to a 111,848 frame dataset composed of 37 different motion capture session for a total of about 16 minutes of motion. Our test datasets do not currently mix data from multiple individuals. Some of the collections we use for testing contain a single activity: an actor shadow boxing, performing various exercises, or walking around in an erratic fashion. Other collections contained sequences representing different kinds of motions. All the animations were taken from the CMU Motion Capture Database [CMU] and use a sample rate of 120 Hz. Some motions contain pose errors or pose noise that is typical of uncleaned motion capture data, and this is noticeable at some points during our graph walks shown in the video for this paper. The system is implemented in MATLAB. The given timings are for an Intel Pentium D 3.0 GHz machine with 1 Gb of memory.

Running vector quantization requires 17.7 seconds on a 14,654 sample dataset (200 clusters) up to 486 seconds on the largest dataset (2000 clusters). For these two examples, string motif extraction requires 95.6 seconds and 358 seconds, respectively. Finally, converting to motion motifs requires 16.3 seconds and 59.7 seconds, respectively, most of this being spent performing dynamic time-warping. The total time required to extract a motion-motif graph is therefore around 15 minutes for the 16 minute dataset.

The final motion-motif structure needs very little space, requiring only pointers to the starting and ending frames for each motion subsequence in a motion motif. The total size of the motion motifs for the largest dataset uses about 100 KB of disk space in an uncompressed text format.

The content of a motion motif is difficult to illustrate in printed form, and the reader is encouraged to view the video associated with this paper, where we display results for two different datasets. Figure 4 visualizes two of the motion motifs extracted for the largest dataset by showing the middle frame for all the motions contained in the motifs. The motion motif on the left contains 8 jumping jack motions ranging in size from 106 to 147 frames, the motion motif on the right contains 14 motions ranging in size from 65 to 127 frames. The original 16 minute dataset corresponds to all the motion capture sessions of subject 14 in the CMU Motion Capture Database [CMU]. It portrays a wide array of activities, including shadow boxing moves, exercising, sweeping the floor, and various other everyday activities.

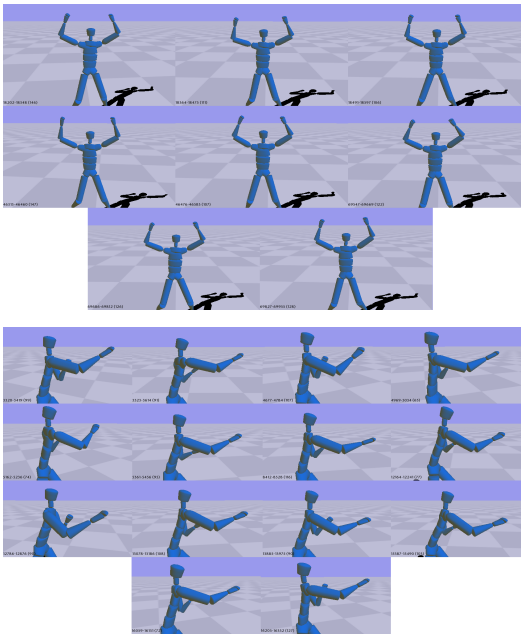

Figure 4: *Content of two discovered motion motifs. The middle frame is shown for 8 motions of a jumping jack motif and 14 motions of a right-punch motif.*

The entire dataset was partitioned in 138 motifs, including the two displayed in Figure 4. The average number of motion per motif is 3.86 while the average number of frame per motion is 139.5. In total, 71230 frames are included within motifs, representing 64% of the entire dataset. We believe that this last number, which we call the percentage of coverage, clearly highlights our goal of partitioning the entire dataset rather than identifying only the longest or most frequently occuring motifs.

Another important aspect relates to the similarity of the motions within a motif. To evaluate this we calculate the average frame error, which we define as the mean euclidean distance between the frames in the motif and the corresponding frame in the averaged motion. For the 16 minutes dataset, the average error is 6.1. As a comparison point, the mean distance between a pose and the average pose of the dataset is 27.0. The error computation is performed in the 18 dimensional pose space resulting from PCA projection.

5.2. Parameter Selection

For all the results presented in this paper we use $\rho = 1$, $\gamma = 1.5$, $\lambda^- = 5$, $\lambda^+ = 30$. The number of clusters used during vector quantization, on the other hand, has to be adjusted based on the size and the complexity of the dataset. For the 16 minutes dataset we use $\kappa = 2000$. We use $\kappa = 200$ for two smaller datasets, the first one a 2.5 minutes dataset containing 58 different sequences of a character walking, running and jumping; the second a 2 minute dataset containing various exercising motions.

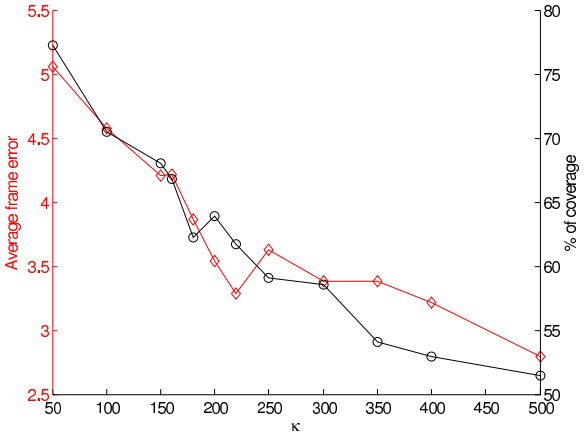

Figure 5: *Effect of* κ*, the number of pose clusters, on the average frame error (*$\diamond$*, left scale) and the percentage of coverage (*$\circ$*, right scale). Data is for* $\rho = 1$.

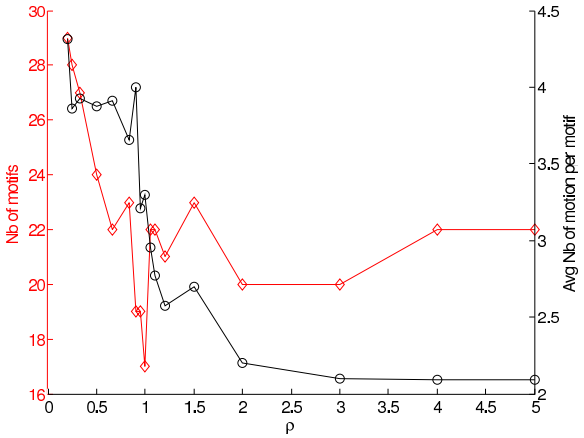

Figure 6: *Effect of* ρ *on the total number of motifs (*$\diamond$*, left scale) and average number of motions per motif (*$\circ$*, right scale). Data is for* $\kappa = 200$.

We studied the effect of the two most important parameters, κ and ρ, on the exercising dataset. The results are presented in Figures 5 and 6. The algorithm was run once for every point shown in the graphs. The inherent noise apparent in these plots is partly due to the randomness during K-means initialization and seed substring selection.

Figure 5 shows that, as κ increases, the average frame error drops rapidly, up to a point where it levels. On the other hand, the percentage of coverage also decreases. We should therefore select κ as low as possible in order to partition a large a portion of the dataset while maitaining a reasonable error. In practice, we obtained very good results with κ rang-

ing from 150 to 300. Values of κ slightly above or below this range still generate acceptable results.

Parameter ρ seems to have a limited impact on the percentage of coverage and the average frame error. For values of ρ between 0.2 and 5, the percentage of coverage varies from 61% to 65% and the average error oscillates between 3.2 and 3.9. Although ρ has little impact on the quality of the segmentation, as measured by these values, it does affect the structure of the resulting graph. Figure 6 shows that, as ρ increases, the average number of motions per motif decreases. However, motifs associated with a large number of motions usually correspond to nodes of the graph that have a large branching factor, and would therefore seem desirable. However, small values of ρ also tend to generate many motifs, resulting in a larger graph that can fail to capture longer, semantically meaningful motions. In practice, we have observed that values of ρ in the neighborhood of 1 tend to produce rich graphs that capture most of the important longer structured motions.

In practice the randomness involved in the algorithm did not seem to affect the results negatively. Similar results were consistently obtained when running the algorithm multiple times with different random seeds.

5.3. Motion-Motif Graphs

Figure 7 presents the graph resulting from the 2 minute exercising dataset. The layout has been performed automatically by GraphViz. The nodes are displayed as rectangles with height proportional to the number of motion in the motif and width proportional to the number of frames per motion. Nodes containing semantically meaningful motions have been manually labeled. The content of a number of nodes is shown in the accompanying video. The graph embeds all the motions from the source motion capture dataset. The majority of the motion is incorporated into motion-motif nodes, as indicated by the percentage of coverage. Frames that are not part of a motion motif correspond to transition motions and appear as edges in the graph.

The motion-motif graph for the 16 minutes dataset is presented in Figure 8. The graph layout has been performed automatically by GraphViz. The large number of nodes in this graph illustrates the wide variety of motions present in the dataset. Exploring this graph allows us to observe that neighboring nodes and strongly connected regions usually contain motions from a similar activity. Edges leaving these strongly connected regions correspond to transitions between different types of motions. These transitions arise either from being explicitly observed in the dataset, or else from the similarity between two motions belonging to different activities. Various disconnected components are also visible, illustrating the fact that no transitions between the corresponding activities have been observed in the dataset. For example, this dataset does not include a transition from drinking to boxing.

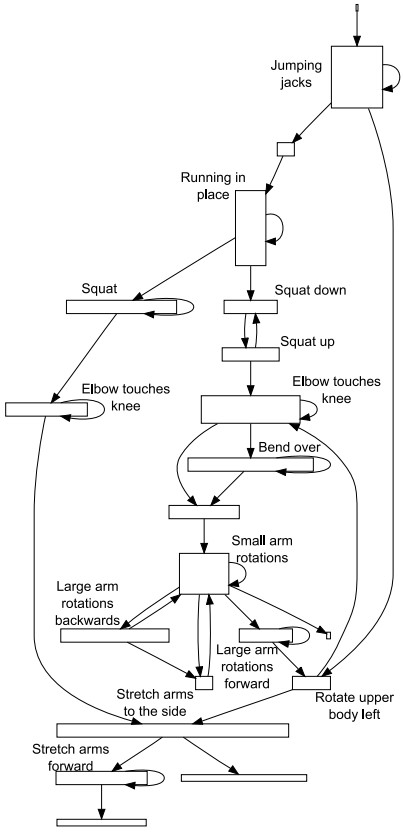

Figure 7: *Motion-motif graph extracted from a 14,654 frame exercising dataset.*

Some nodes of this graph are displayed in the accompanying video.

Lastly, we tested the extracted graphs for resequencing by observing the output for various directed walks through the graphs. The results are shown in the accompanying video. Some minor motion artifacts remain because we do not correct for effects such as foot skate during blending.

Since the edges of motion-motif graphs only correspond to transitions that have actually been observed in the data, they are often too sparsely connected for resequencing applications that require fast response times. In such cases the graph connectivity could easily be enriched by automatically detecting good transition points that occur *within* a motif, in the vein of standard motion graph techniques. The motion-motif graph still serves a variety of other purposes, not the least of which is providing an understandable view into the structure of the motion dataset.

5.4. Additional Results

We have verified that the extracted motion-motif graphs are relatively stable when band-limited Gaussian noise is added

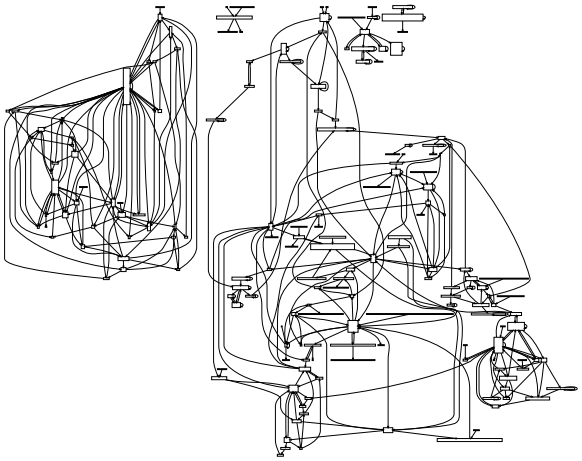

Figure 8: *Motion-motif graph extracted from a* 111*k frame dataset.*

to the joint angles. We have also demonstrated that the discovered motion motifs can be effectively compressed using a PCA-like scheme that directly uses motion instances to define compact non-orthogonal motion bases. This allows good reconstruction of the original motion instances that comprise a bundle while obviating the need to additionally store large PCA reconstruction matrices. This format also directly supports linearly weighted blending and interpolation. In the interest of space, we point the reader to [Bea07] for noise robustness tests, the details of the motion motif compression scheme, and motion-motif graphs of additional datasets.

6. Conclusion

We have presented a technique that can distill a highly structured representation from large unstructured motion capture datasets. The resulting motion-motif graph provides a visualization of the motion data, its repetitive variations, and its connectivity. The user can specify a preference for the desired shape of the extracted motion motifs. The processed result is multi-purpose, potentially usable for motion resequencing, creating blend spaces, motion compression, motion segmentation, and motion annotation. A key insight of the method is to tackle segmentation and clustering in a coupled fashion with a preference given to large-volume motion motifs. We note that there is no guarantee that motion motifs will always be semantically meaningful, nor can they necessarily make the fine distinctions that might be needed to discriminate between two very similar classes of motion.

Finding rich and complex structure in multivariate time series data usable for both analysis and synthesis of character animation is an open ended problem. Aside from transition blending and compression, we have not yet explored other applications that the continuous parameterizations of the motions within a motifs would afford. With some optimizations, an online version of the algorithm could be used to support interactive graph construction. We wish to develop data structures that will enable the technique to scale to hundreds of hours of motion capture data instead of the current tens of minutes.

7. Acknowledgements

This work was supported in part by grants from NSERC and FQRNT. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

References

[AAT*06] ARAKI Y., ARITA D., TANIGUCHI R., UCHIDA S., KURAZUME R., T. H.: Construction of symbolic representation from human motion information. In *Lecture Notes in Computer Science*. Springer, 2006, pp. 212–219.

[AF02] ARIKAN O., FORSYTH D.: Interactive motion generation from examples. *ACM Trans. Graphics 21*, 3 (2002), 483–490.

[BE94] BAILEY T., ELKAN C.: Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Second Int'l Conf. on Intelligent Systems for Molecular Biology* (1994), pp. 28–36.

[Bea07] BEAUDOIN P.: *Compression de données d'animation acquises par capture de mouvements*. PhD thesis, Université de Montréal, 2007.

[BH00] BRAND M., HERTZMANN A.: Style machines. In *Proc. SIGGRAPH '00* (2000), pp. 183–192.

[BSC05] BASU S., SHANBHAG S., CHANDRAN S.: Search and transitioning for motion captured sequences. In *VRST '05: Proc. Symp. Virtual Reality Software and Technology* (2005), pp. 220–223.

[BSP*04] BARBIČ J., SAFONOVA A., PAN J.-Y., FALOUTSOS C., HODGINS J., POLLARD N.: Segmenting Motion Capture Data into Distinct Behaviors. In *Proc. Graphics Interface* (July 2004), pp. 185–194.

[CKL03] CHIU B., KEOGH E., LONARDI S.: Probabilistic disovery of time series motifs. In *Conf. on Knowledge Discovery in Data* (2003), pp. 493–498.

[CMU] CMU: Carnegie melon university graphics lab motion capture database.

[FF05] FORBES K., FIUME E.: An efficient search algorithm for motion data using weighted pca. In *SCA '05: Proc. Symp. Computer Animation* (2005), pp. 67–76.

[FMJ02] FOD A., MATARIC M., JENKINS O.: Automated derivation of primitives for movement classification. *Autonomous Robot 12*, 1 (2002), 39–54.

[HG07] HECK R., GLEICHER M.: Parametric motion graphs. In *Proc. Symp. Interactive 3D Graphics and Games* (2007).

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

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

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

[KS05] KWON T., SHIN S.: Motion modeling for on-line locomotion synthesis. In *SCA '05: Proc. Symp. Computer Animation* (2005), pp. 29–38.

[LCR*02] LEE J., CHAI J., REITSMA P., HODGINS J., POLLARD N.: Interactive control of avatars animated with human motion data. *ACM Trans. Graphics 21*, 3 (2002), 491–500.

[LKLC03] LIN J., KEOGH E., LONARDI S., CHIU B.: A symbolic representation of time series, with implications for streaming algorithms. In *Proc.8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (2003), pp. 2–11.

[LM06] LIU G., MCMILLAN L.: Segment-based human motion compression. In *SCA '06: Proc. Symp. on Computer Animation* (2006), pp. 127–135.

[LWS02] LI Y., WANG T., SHUM H.-Y.: Motion texture: a two-level statistical model for character motion synthesis. *ACM Trans. Graphics (Proc. SIGGRAPH) 21*, 3 (2002), 465–472.

[LZWM05] LIU G., ZHANG J., WANG W., MCMILLAN L.: A system for analyzing and indexing human-motion databases. In *SIGMOD '05: Proc. ACM SIGMOD Intl. Conf. on Management of Data* (2005), pp. 924–926.

[MIES07a] MINNEN D., ISBELL C., ESSA I., STARNER T.: Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery. *IEEE Int. Conf. on Data Mining* (2007).

[MIES07b] MINNEN D., ISBELL C., ESSA I., STARNER T.: Discovering Multivariate Motifs Using Subsequence Density Estimation and Greedy Mixture Learning. *Proc. AAAI 22*, 1 (2007), 615.

[MR06] MÜLLER M., RÖDER T.: Motion templates for automatic classification and retrieval of motion capture data. In *SCA '06: Proc. Symp. Computer Animation* (2006), pp. 137–146.

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

[MSEI07] MINNEN D., STARNER T., ESSA I., ISBELL C.: Improving Activity Discovery with Automatic Neighborhood Estimation. *Int. Joint Conf. on Artificial Intelligence (IJCAI07)* (2007), 6–12.

[MWS*06] MINNEN D., WESTEYN T., STARNER T., WARD J., LUKOWICZ P.: Performance Metrics and Evaluation Issues for Continuous Activity Recognition. *Performance Metrics for Intelligent Systems* (2006).

[MYHW08] MENG J., YUAN J., HANS M., WU Y.: Mining Motifs from Human Motion. In *Eurographics 2008 - Short Papers* (2008), pp. 71–74.

[Oat02] OATES T.: PERUSE: An unsupervised algorithm for finding recurring patterns in time series. In *Intl Conf. on Data Mining* (2002), pp. 330–337.

[PRM00] PAVLOVIC V., REHG J., MACCORMICK J.: Learning switching linear models of human motion. In *NIPS '00: Proc. Neural Information Processing Systems* (2000), pp. 981–987.

[PSKS04] PARK S., SHIN H., KIM T., SHIN S.: On-line motion blending for real-time locomotion generation. *Comput. Animat. Virtual Worlds 15*, 3-4 (2004), 125–138.

[RPE*05] REN L., PATRICK A., EFROS A., HODGINS J., REHG J.: A data-driven approach to quantifying natural human motion. *ACM Trans. Graphics 24*, 3 (2005), 1090–1097.

[SB05] SO C., BACIU G.: Entropy-based motion extraction for motion capture animation. *Comput. Animat. Virtual Worlds 16*, 3-4 (2005), 225–235.

[SB06] SO C., BACIU G.: Hypercube sweeping algorithm for subsequence motion matching in large motion databases. In *VRCIA '06: Proc. ACM Intl. Conf. Virtual Reality Continuum and its Applications* (2006), pp. 221–228.

[SO06] SHIN H., OH H.: Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proc. Symp. Computer Animation* (2006), pp. 291–298.

[TH00] TANCO L., HILTON A.: Realistic synthesis of novel human movements from a database of motion capture examples. In *HUMO '00: Proc. Workshop on Human Motion* (2000), pp. 137–142.

[TIU05] TANAKA Y., IWAMOTO K., UEHARA K.: Discovery of time-series motif from multi-dimensional data based on mdl principle. *Machine Learning 58*, 2-3 (2005), 269–300.

[YKM*07] YANKOV D., KEOGH E., MEDINA J., CHIU B., ZORDAN V.: Detecting time series motifs under uniform scaling. *Proc. 13th ACM SIGKDD Intl Conf on Knowledge Discovery and Data Mining* (2007), 844–853.