Volume xx (200y), Number z, pp. 1–10

Finding Syntactic Structures from Human Motion Data

Jong Pil Park[1] Kang Hoon Lee[2] Jehee Lee[1]

[1]Seoul National University
[2]Kwangwoon University

Abstract

We present a new approach to motion rearrangement that preserves the syntactic structures of an input motion automatically by learning a context-free grammar from the motion data. For grammatical analysis, we reduce an input motion into a string of terminal symbols by segmenting the motion into a series of subsequences, and then associating a group of similar subsequences with the same symbol. In order to obtain the most repetitive and precise set of terminals, we search for an optimial segmentation such that a large number of subsequences can be clustered into groups with little error. Once the input motion has been encoded as a string, a grammar induction algorithm is employed to build up a context-free grammar so that the grammar can reconstruct the original string accurately as well as generate novel strings sharing their syntactic structures with the original string. Given any new strings from the learned grammar, it is straightforward to synthesize motion sequences by replacing each terminal symbol with its associated motion segment, and stitching every motion segment sequentially. We demonstrate the usefulness and flexibility of our approach by learning grammars from a large diversity of human motions, and reproducing their syntactic structures in new motion sequences.

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

1. Introduction

Since the advent of motion capture technology, synthesizing realistic animation sequences by using captured motion data has been an active research topic in computer graphics. Beyond just playing back captured data kept intact, there are currently various approaches for creating a large range of variations on existing motion data. One successful approach among those is to rearrange captured motion frames into new sequences in the time domain. Its primary benefit is that arbitrarily long animation sequences can be synthesized from an input motion data of a limited size, which currently plays an essential role in creating interactively controlled characters and animation authoring systems based on motion database.

The basic requirement for motion rearrangement is that any successive frames of the resulting motions should change smoothly. This condition is often efficiently satisfied by preprocessing original motion data into a transition graph that encodes the connectivity among similar frames, and then finding paths on the graph. This graph representation is so powerful that any random paths correspond to novel smooth sequences. However, we often need higher-level constraints than just the smoothness of motion to generate reasonable sequences of actions that can be perceived as meaningful activities.

For example, many dance genres have their own repertoires of basic steps as well as typical patterns that sequence those steps in a natural way. If someone imitates a professional dancer by just copying partial steps and combining those in an arbitrary order, his/her performance would be hardly recognized as a well-formed dance. Similarly, motion synthesis based on captured data would fail in reproducing such an activity if its basic elements and their combinatorial patterns are not preserved as constraints.

We present a new approach to motion rearrangement based on grammatical analysis of motion data for identifying elementary actions and learning a set of rules to form reason-

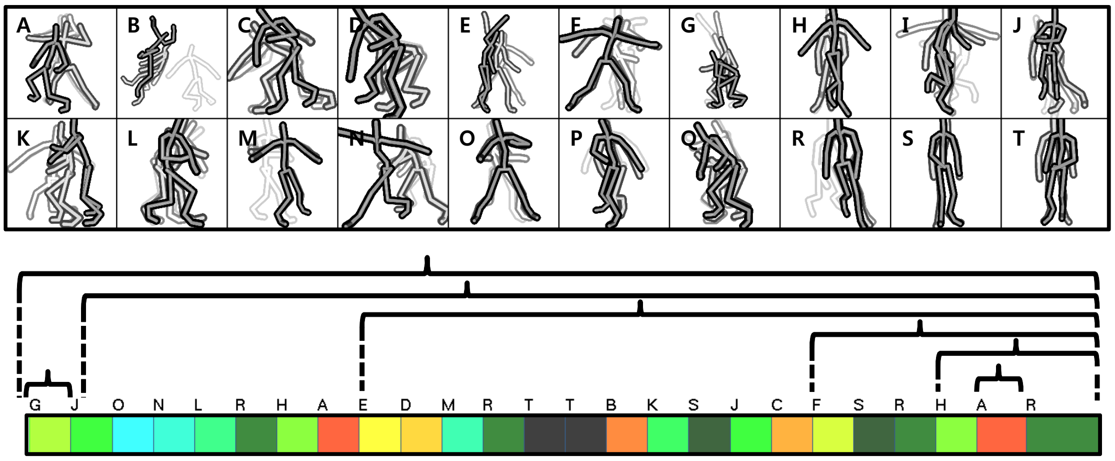

Figure 1: *The overview of our approach. (Top) We identified a set of terminal symbols from the break-dancing motion data. Each terminal symbol represents an elementary action, and is associated with a set of subsequences of the input motion. (Bottom) The input motion is encoded as a string of terminal symbols by replacing each segmented interval with its associated symbol. The horizonal bar at the bottom visualizes a substring of the encoded string by depicting the subsequences belonging to each symbol with the same color. We learn a context-free grammar from a set of substrings so that each substring can be parsed compactly with the grammar.*

able sequences of those actions (see Figure 1). Each elementary action is represented as a terminal symbol of our grammar at which a set of subsequences of input motion are associated. We formulate the task of identifying terminal symbols as an optimization process that searches for an optimal segmentation of input motion such that the segmented intervals can be clustered into a given number of symbols most abundantly and precisely. Once an input motion has been encoded as a string of terminal symbols, we learn a context-free grammar from the string by using a grammar induction method developed in computational linguistics. The learned grammar can not only reconstruct the original string accurately, but also generate novel strings that preserve the syntactic structures of the original string. For arbitrary new strings produced by our grammars, we can synthesize their corresponding motion sequences simply by replacing each terminal symbol with one of its associated subsequence, and stitching such subsequences sequentially.

The key advantage of our approach is that complex hierarchical patterns of human movement can be automatically learned from unlabelled motion data. When synthesizing new motions from learned grammars, no additional constraints are necessary for preserving the original syntactic structures. Our approach is ideally suited for well-formed physical activities such as sports and dancing, whose embedded patterns are often difficult to be recognized comprehensively and be reproduced accurately. To the best of our knowledge, this work provides the first method of data-driven motion analysis and synthesis based on context-free grammars.

We demonstrate the usefulness and flexibility of our approach by learning grammars from a large diversity of human motions including locomotion, basketball, break-dancing, and boxing. The significant syntactic structures governing those motions are maintained well in new motion sequences produced by the learned grammars. To support this assertion, we perform the visual comparison between our results and the motions synthesized by the graph-based motion rearrangement.

2. Background

The ability of capturing three-dimensional motion has not only contributed to the understanding of human movement for science and medicine, but also opened a new way of expressing highly realistic motion within virtual worlds for computer games and movies. In order to express a large diversity of motions using a limited amount of motion data, many useful methods have been investigated in computer graphics. Currently, we are able to edit a motion sequence to satisfy geometric constraints [LS99], blend multiple sequences spatially into an intermediate motion [KG03], stitch multiple sequences temporally into a long sequence [LCR*02, AFO03], and take several of these effects simultaneously [SO06, HG07, SH07]. Based on these basic capabilities, we can create high-level motion controllers that fully utilize available motion data to synthesize natural and rich animation of virtual characters [LL06, TLP07].

Data-driven motion synthesis often requires input motion data be pre-processed into computationally efficient and easily recognizable structures. Partitioning motion data into a series of subsequences is a typical process for obtaining meaningful movement intervals [BSP*04, FMJ02, KS05]. Furthermore, classifying several motion intervals according to their similarity provides a higher-level view of motion data as a combination of elementary actions. Measuring the similarity between two motions typically involves pose-to-pose distance calculation. The distance evaluation between two skeletal poses has been addressed by several different approaches based on joint angles [LCR*02], point clouds [KGP02], geometric relational characteristics [MRC05], and dimensionally reduced representation [FF05].

For large motion database, retrieving a set of similar motion intervals efficiently is a crucial problem. Kovar et al. presented a match web structure for searching over a large motion database to identify a set of similar motions efficiently [KG04]. Liu et al. indexed motion sequences compactly based on a locally linear model describing groups of similar poses [LZWM05]. Müller et al. introduced the concept of motion templates that compactly characterize the essence of similar motions [MR06]. Meng et al. precomputed the best matches for each frame in a compact structure that supports fast motion retrieval [MYHW08]. Deng et al. achieved high perceptual consistency in motion retrieval based on a hierarchical representation of motions [DGL09]. Recently, Krüger et al. developed a fast search method based on kd-tree structure [KTWZ10].

Learning grammatical structures from visual data has been focused recently in computer graphics. Št'ava et al. presented an inverse procedural modeling method to automatically generate L-systems, a kind of context-free grammars, that represent an input two-dimensional model [vBM*10]. With the same spirit, Bokeloh et al. developed a method of building a context-free grammar from three-dimensional geometric models based on their previous work on partial symmetry of shapes [BWS10]. For human motion, Beaudoin et al. introduced a motion-motif graph providing understandable structure to the contents and connectivity of input motion data [BCvdP08]. In computational linguistics, grammar induction (or grammatical inference) has been an active research topic, which aims at learning a formal grammar automatically from a corpus of texts [dlH05]. We employ the alignment-learning algorithm among many induction methods due to its efficiency and freely available implementation [vZ00].

3. Overview

Our method receives an unlabelled human motion sequence $M = \{\mathbf{x}_t | 1 \le t \le T\}$ as its input, where $\mathbf{x}_t$ denotes a skeletal pose at an instance, and learns a context-free grammar that describes the syntactic structures of the input sequence. The grammar G is represented as a set $\{V, \Sigma, R, S\}$, where V and Σ are sets of non-terminal and terminal symbols, respectively, S is the start symbol belonging to V, and R is a set of production rules. In context-free grammars, the production rules have a form of $X \rightarrow \gamma$, where $X \in V$ is a non-terminal, and γ is a sequence of terminals and non-terminals that can rewrite X.

Learning G begins with identifying the set of terminal symbols Σ (see Section 4). We intend that each terminal $\sigma \in \Sigma$ is associated with a set of subsequences of the input motion that are close to each other and dissimilar to other subsequences. Given the number of terminal symbols K to be identified, we search for an optimal segmentation of M such that its segmented intervals are clustered into K groups compactly and precisely. Under an objective function that measures the desirable properties for segmentation, we perform a two-level optimization process, in which the genetic algorithm is employed first to find an optimal solution at a coarse resolution, and then this initial guess is finely tuned by using the simulated annealing approach. The K groups obtained from the eventual segmentation are labelled as unique terminal symbols $\{\sigma_1, ..., \sigma_K\}$. Because every segment interval associated with each group shares the same symbol, the entire motion data can be rewrited as a string of terminal symbols $\mathbf{z} = \sigma_{\mathrm{k}(1)}\sigma_{\mathrm{k}(2)} \cdots \sigma_{\mathrm{k}(n)}$, where $\mathrm{k}(i)$ denotes the index of the group to which i-th subsequence belongs.

Training the remaining sets V, R, and S corresponds to building up parse trees converging to S over given strings (see Section 5). If we construct a single parse tree over the entire string $\mathbf{z}$, the grammar can reproduce only the given string. Instead, we partition $\mathbf{z}$ into a consecutive series of substrings $\mathbf{z}_1\mathbf{z}_2 \cdots \mathbf{z}_h$, and build up a parse tree for each substring such that those trees can share as many syntactic structures as possible. The more the substrings share syntactic structures, the more compact and flexible our grammar is. The induction of a grammar from a corpus of example strings has been an important research topic in computational linguistics. However, the partitioing of a string has not been dealt with seriously because written texts are already well partitioned into sentences. For grammar induction, we simply employ an existing method known as the alignment-based learning (abbreviated as ABL) [vZ00], and mainly focus on finding an optimal partitioning of the given string. To do so, we define its objective function based on the principle of the minimal description length, and then generate a set of candidate solutions satisfying some constraints, from which the best one is selected.

Once we have built G, we can reconstruct the input string $\mathbf{z}$ accurately from G by generating a sequence of terminal symbols for each substring in the reverse order of building up its parse tree and concatenating every substring into a single string. In fact, the input string is one special instance that can be obtained from G, and applying rules in new orders produce new arrangements of terminal symbols that preserve the syntactic structures of the original string. The synthesis of a motion sequence from an arbitrary new string is straightforward by stitching a series of motion segments that are associated with the terminal symbols belonging to the string sequentially (see Section 6).

4. Identifying Terminal Symbols

We begin with translating input motion data into a sequence of terminal symbols, e.g. a string, by segmenting the input motion into a series of subsequences, and then associating similar subsequences with the same symbols. Such a symbolic representation allows each symbol to occur repetitively throughout the entire string, and to form various combinatorial patterns with its adjacent symbols, which is an important condition for learning a rich set of grammar rules. However, increasing the repetitiveness of a symbol is often accompanied by decreasing its clarity because more diverse movements can be labelled as the same symbol. In order to achieve a reasonable balance between the repetitiveness and the clarity, we formulate the symbolic encoding of input motion data as an optimization problem based on an objective function of evaluating those two desirable features (see Section 4.1).

A key challenge in the optimization is the size of search space that is growing exponentially with respect to the length of input motion. Assuming that the number of motion frames is T, the number of possible segmentation is 2^{T-1}, because every pair of consecutive frames can be split or not. Also, the number of possible labeling of n subsequences is K^n where K is the number of available symbols. Even if we employ some pruning policies such as enforcing the minimum length of subsequences, it would not be feasible to exhaustively enumerate every possible segmentation and search for an optimal labeling for each segmentation to maximize the objective function.

For more practical solution, we restrict the solution space by allowing only the segmentation to vary in optimization, and forcing the result of labeling to be uniquely determined by given segmentation according to the *PAM* (partitioning around medoids) algorithm [LK90]. Within this reduced space, we perform a two-level heuristic search that first finds an optimal segmentation at a coarse resolution using an existing method based on the genetic algorithm (see Section 4.2), and then refines the coarse segmentation locally using our new method based on simulated annealing (see Section 4.3).

4.1. Scoring Segmentations

Given the number of symbols K and the maximum number of subsequences in a segmentation N, our search space is composed of every possible segmentation $\{\mathbf{s}_1 = [1,t_1), \mathbf{s}_2 = [t_1,t_2), ..., \mathbf{s}_n = [t_{n-1},T]\}$ where the number of segments n is allowed to have an arbitrary value within the range of $[K,N]$. The lower boundary condition $n \geq K$ guarantees that every symbol is associated with at least a single segment. On the other hand, the upper boundary condition is to limit the size of search space. For an arbitrary segmentation satisfying these contraints, we evaluate its score by first clustering its segmented intervals into K groups based on their motion similarity, and then measuring the quality of clusters based on our objective function.

Motion similarity. The similarity between any two segment intervals $[t_s,t_e]$ and $[t'_s,t'_e]$ is measured by aligning two intervals temporally and evaluating the pose-to-pose distance function $\mathrm{d}(\mathbf{x}_t, \mathbf{x}_{t'})$ by [LCR*02] for every matched pair of frames. For efficiency in time alignment, we use a simple uniform time warping such that the m-th frame of the longer interval corresponds to the $\lfloor ml'/l \rfloor$-th frame of the shorter one, where l and l' are the lengths of the longer and shorter intervals, respectively. If the matched pairs of frames after alignment are $\{(t_m,t'_m) | 1 \leq m \leq L = max(l,l'), t_s \leq t_m \leq t_e, t'_s \leq t'_m \leq t'_e\}$, then our motion distance is evaluated as follows.

$$\mathrm{D}([t_s,t_e],[t'_s,t'_e]) = \frac{1}{L}\sum_{m=1}^{L} \mathrm{d}(\mathbf{x}_{t_m}, \mathbf{x}_{t'_m}) \tag{1}$$

Clustering. We use the *PAM* algorithm to parition given segment intervals into a set of clusters $\mathbf{C} = \{\mathbf{C}_k | k = 1, \cdots, K\}$, where each cluster $\mathbf{C}_k$ is associated with a collection of segments $\{\mathbf{s}^i_k | 1 \leq i \leq N_k\}$. The overall process of the *PAM* is almost equivalent to the Lloyd's algorithm for the k-means clustering problem except that the *PAM* allows more general distance measures other than the Euclidean distance. Specifically, the *PAM* minimizes the within-cluster sum of distances as follows.

$$\mathrm{E}(\mathbf{C}) = \sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathrm{D}(\mathbf{s}^i_k, \mathbf{s}^c_k) \tag{2}$$

where $\mathbf{s}^c_k$ represents the central segment (i.e. medoid) of the k-th cluster. As the initial guess of medoids for the *PAM*, we select always the first K segments to guarantee that the clustering results are uniquely determined by given segmentations. We label each k-th cluster with a unique symbol σ_k, which constructs a set of terminal symbols $\{\sigma_k | k = 1, \cdots, K\}$.

Quality of clusters. The repetitiveness of symbols is evaluated indirectly by calculating the ratio of the total lengths of central segments $L(\mathbf{s}^c_k)$ to the entire length of input motion data T for obtaining a normalized measure within the range of $[0,1]$ as follows.

$$\mathrm{R}(\mathbf{C}) = \frac{1}{T}\sum_{k=1}^{K} L(\mathbf{s}^c_k) \tag{3}$$

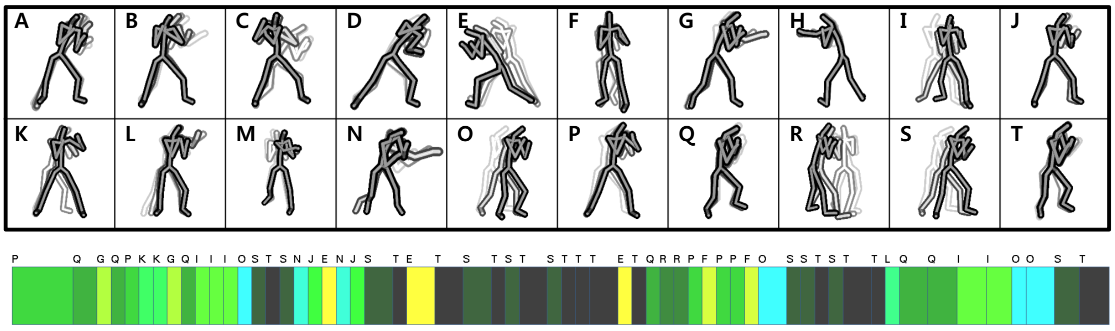
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
P Q G Q P K K G Q I I I O S T S N J E N J S T E T S T S T S T T T E T Q R R P F P P F O S S T S T T L Q Q I I O O S T

Figure 2: *The string representation of boxing motion data and its terminal symbols.*

For a fixed length of input motion, low $\mathrm{R}(\mathbf{C})$ means that symbols are associated with a large number of short subsequences, which corresponds to high repetitiveness. As its opposite extreme case, the maximum $\mathrm{R}(\mathbf{C})$, e.g. 1, is achieved when each symbol is associated with exactly one subsequence, which corresponds to no repetition of symbols.

The clarity of symbols represents how similar movements are associated with the same symbols such that each symbol can be clearly matched to a distinctive movement type. The within-cluster sum of distances $\mathrm{E}(\mathbf{C})$ in Equation 2 is a feasible measure for this feature, but need to be slightly modified to restrict its range to $[0,1]$ without respect to the length of input motion and the diversity of poses as follows.

$$\mathrm{E}'(\mathbf{C}) = \frac{1}{\mathrm{d}_{max}} \sum_{k=1}^{K} \sum_{i=1}^{N_k} \frac{L(\mathbf{s}_k^i)}{T} \mathrm{D}(\mathbf{s}_k^i, \mathbf{s}_k^c) \tag{4}$$

where d_{max} is the maximum distance among the distances of every pair of two poses $\mathrm{d}(\mathbf{x}_t, \mathbf{x}_{t'})$. We take the inverse of the weighted sum of both features as our objective function in order to generate higher score for lower sum of R and E$'$, where the weight ω controls the relative contributions to the final score $(0 \le \omega \le 1)$.

$$\mathrm{F}(\mathbf{C}) = \frac{1}{\omega \mathrm{R}(\mathbf{C}) + (1-\omega)\mathrm{E}'(\mathbf{C})} \tag{5}$$

4.2. Coarse Segmentation

We employ an existing segmentation method based on the genetic algorithm by [LPL11], which is briefly summarized in this section. To apply the genetic algorithm for combinatorial optimization problems, the solution domain typically needs to be represented as a linear array of binary values. For the problem of segmenting an input motion of T frames, this requirement can be easily accomplished by interpreting each binary value b_i in an array $b_1 b_2 ... b_n$ as whether to select or not the i-th cutting location among a set of candidate locations $\mathbf{t} = \{t_1, t_2, ..., t_n\}$. Setting $\mathbf{t}$ as every inbetween frame $\{1, 2, ..., T-1\}$ enables searching over the finest resolution of segmentation, but produces impractical size of search space even for moderate lengths of input motion. Instead, we subsample the finest set at regular intervals, e.g. 30 frames, to search over a coarser resolution of segmentation.

In optimization, an initial population of solutions (e.g. binary arrays) is randomly generated, and then iteratively evolves into next generations until the solutions converge or the number of iterations reaches a pre-defined maximum. At each generation, each solution is evaluated by our objective function in Equation 5, and reproduced in the next generation probablistically according to its score. The best solution at the last generation is selected as the initial coarse segmentation.

4.3. Fine Tuning

We improve the quality of the initial guess by performing a simulated annealing search so that segment intervals can be gradually adjusted to increase the score of the entire segmentation without being restricted to predefined cutting locations in the earlier stage. For each iteration of the search process, we select an arbitrary segment $\mathbf{s}_i = [t_i, t_{i+1})$ from the current segmentation of n segments. Then, we locally adjust the segmentation around $\mathbf{s}_i$ by carrying out one of the following two operations.

- **Slide** translates $\mathbf{s}_i$ into a new interval of $[t_i \pm d, t_{i+1} \pm d)$, which in turn resizes its adjacent segments $\mathbf{s}_{i-1}$ and $\mathbf{s}_{i+1}$ into $[t_{i-1}, t_i \pm d)$ and $[t_{i+1} \pm d, t_{i+2})$, respectively.
- **Resize** extends or narrows $\mathbf{s}_i$ into a new interval of $[t_i, t_{i+1} \pm d)$, which in turn resizes its next segment $\mathbf{s}_{i+1}$ into $[t_{i+1} \pm d, t_{i+2} \pm d)$.

For each operation on a selected segment, we also need to keep the clusters consistent with the resulting segmentation. This involves the removal of old intervals, the insertion of new intervals, and the update of medoids for expanded or shrunken clusters. For example, resizing an interval $\mathbf{s}_{old} = [t_a, t_b)$ into $\mathbf{s}_{new} = [t_a, t_b + d)$ leads to the resize of its next segment $\mathbf{s}'_{old} = [t_b, t_c)$ into $s'_{new} = [t_b + d, t_c)$. Then, we need to remove both $\mathbf{s}_{old}$ and $\mathbf{s}'_{old}$ from their existing clusters, and insert both $\mathbf{s}_{new}$ and $\mathbf{s}'_{new}$ into their closest clusters determined by the distances to the current medoids $\{\mathbf{s}^c_k\}$. For every cluster affected by those insertions and removals, we update its medoid to a new one that has the minimal sum of distances to every other segment belonging to the same cluster. Finally, the calculation of Equation 5 with the new segmentation and clustering reveals whether the applied operation has created a better categorization or not in comparison with the latest value of the objective function. If the operation has made a better result, we accept the new categorization. Otherwise, the segmentation and clustering revert to their previous states.

The parameter d for slide and resize operations controls the degree of variability of segmentation. We use a large d at the initial phase of our optimization for extensively exploring the search space, and decrease the value according to the number of iterations for finely tuning the solution at the latter phase. Our optimization process terminates when the value of d has reached its minimum, and the value of the objective functions has not been lowered for a pre-determined number of iterations.

5. Learning Grammars

Given a string of terminal symbols for input motion data, we basically intend to discover a set of grammar rules that can reconstruct the original string. As one possible approach to this problem, we can search for a single, global parse tree spanning the entire string [NMW97]. Such an approach is highly useful for compressing the string without loss of original data as well as understanding its inherent syntactic structures. However, this approach is limited to reproducing exactly one given string, and is not appropriate for our purpose of using the result of grammatical analysis as a foundation for motion syntheis.

Instead of regarding the entire string as a single instance of a grammar, we take another view that the input string is composed of a sequence of substrings, each of which corresponds to an instance of a grammar. Learning a common grammar from multiple instances allows not only to reproduce each substring correctly, but also to synthesize new arrangements of terminal symbols that at least locally preserve the syntactic structures of the original string. Considering each substring as a sentence in written language, this corresponds to an important research topic in computational linguistics, referred to as grammar induction or grammatical inference. There are a lot of available methods for grammar induction that learns regular or context-free grammars from a corpus of example sentences in an unsupervised manner. Among those, we employ the alignment-based learning (ABL) for its simplicity and efficiency for our experiments [vZ00]. Once we have left the internal process of grammar induction to an existing method, the only remaining problem is to provide the black box with an appropriate input, that is, a set of substrings partitioned from the entire string.

5.1. Evaluating the Quality of Sentences

Under the principle of the minimum description length in information theory, we pursue the most compactly encodable partitioning that minimizes the total length for encoding its resultant grammar and substrings [LS00]. For example, let us consider an input string *abxcdabycdabxcd*. Paritioning the string evenly into *abxcd*, *abycd* and *abxcd*, those can be concisely encoded as a small set of grammar rules $\{S \rightarrow abAcd, A \rightarrow x, A \rightarrow y\}$ and the simple sequence of rules for rebuilding each substring $\{S_1A_1, S_1A_2, S_1A_1\}$, where X_i denotes the i-th rule among the rules whose left-hand side is X. On the other hand, selecting other possibilities of partitioing such as *abxc*, *dabycda* and *bxcd* would yield larger sets of grammar rules and more complex sequences of rules for reconstruction.

In order to quantitatively measure the goodness for any arbitrary sets of substrings $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, ...\}$, we first learn a grammar $\mathbf{G}$ by running an induction method on $\mathbf{Z}$, and then evaluate the total description length $\mathrm{DL}(\mathbf{Z})$ as the number of bits for storing the grammar itself $\mathrm{B}(\mathbf{G})$ in addition to the number of bits for storing the information for rebuilding the given substrings $\hat{\mathrm{B}}(\mathbf{Z}|\mathbf{G})$ as follows.

$$\mathrm{DL}(\mathbf{Z}) = \mathrm{B}(\mathbf{G}) + \hat{\mathrm{B}}(\mathbf{Z}|\mathbf{G}) \tag{6}$$

As one possible way for storing the grammar, we assume that every grammar rule is concatenated into a single sequence of symbols with a separator symbol between each adjacent pair of rules. For example, the rule set in the above example converts into *SabAcd@Ax@Ay*, where @ represents the separator. We approximate the minimal number of bits for each symbol as $\log_2(|V| + |\Sigma| + 1)$ because it may be a non-terminal, a terminal, or the separator. $\mathrm{B}(\mathbf{G})$ is evaluated by counting the number of bits for all symbols in the concatenated sequence.

When encoding each substring, we first build its parse tree based on the grammar, and then traverse the tree in preorder while generating the code for each non-terminal node as the index of the selected rule among every candidate rule that can rewrite the node. If the number of candidate rules for a given node is h, the minimal number of bits for the node is $\log_2 h$. $\hat{\mathrm{B}}(\mathbf{Z}|\mathbf{G})$ is obtained by totalling the number of bits for every substring. For example, the first substring *abxcd* in the above example can be reconstructed by selecting the first

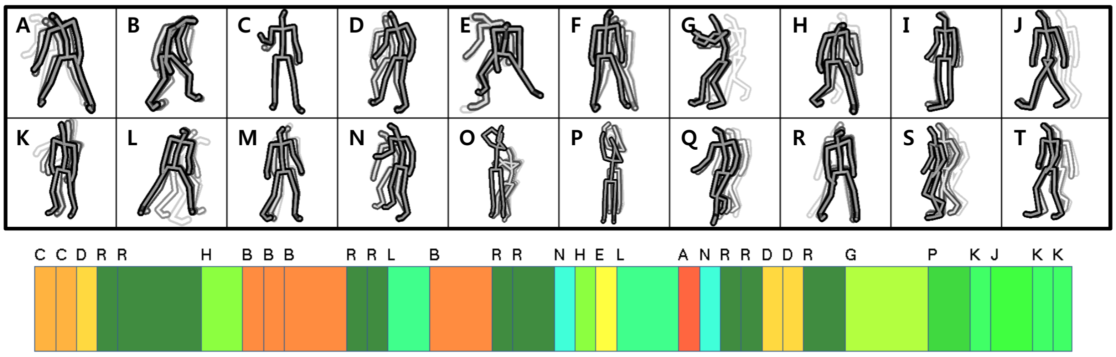
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
C C D R R H B B B R R L B R R N H E L A N R R D D R G P K J K K

Figure 3: *The string representation of basketball motion data and its terminal symbols.*

rule $S \rightarrow abAcd$ for the start symbol S, followed by also the first rule $A \rightarrow x$ for the non-terminal A. Because S has only one candidate rule, the minimal number of bits for storing the first production is 0 ($= \log_2 1$), that is, no data required. On the other hand, the second production needs 1 ($= \log_2 2$) bit since we have two production rules for A. As a result, *abxcd* is encoded simply as a single bit '0'. Two other substrings can be encoded similarly as '1' and '0', respectively. The total number of bits is thus 3 in this example.

5.2. Searching for High-Quality Sentences

Given the evaluation function $\mathrm{DL}(\mathbf{Z})$ over any partitioning of a string, the problem of finding the most compactly encodable one looks much like our earlier problem of identifying terminal symbols except the granularity of segmentation. However, it is not so straightforward just to employ the same approach because $\mathrm{DL}(\mathbf{Z})$ is much more involved than $\mathrm{F}(\mathbf{C})$ measuring the repetitivenss and clarity of frame intervals. First of all, a slight modification of $\mathbf{Z}$ often yields a drastic variation of its grammar. Such behavior eventually makes the whole landscape of $\mathrm{DL}(\mathbf{Z})$ highly oscillatory, which is not an ideal situation for the genetic algorithm as well as the simulated annealing. Furthermore, the evaluation of $\mathrm{DL}(\mathbf{Z})$ requires too much computational cost to be used repeatedly at each iteration in an optimization process till convergence.

Instead of evolving our solution gradually toward an optimal one, we take a simpler approach that randomly generates a number of candidate solutions and picks the best one. Each candidate solution is created by determining a pre-defined number of cut locations over the input string. We impose two constraints upon our solution space to avoid undesirable candidates. The first constraint is simply the minimum length of partitioned substrings. As the second constraint, we prevent our solution from splitting repeatedly occurred substrings such as *abxcd*, *ab* and *cd* in *abxcdabycdabxcd*, because those are highly probable to constitute syntactic substructures. To enumerate every repeated substring, we use an efficient algorithm of $O(N)$ time complexity where N is the length of the input string [Gus97].

6. Motion Synthesis

It is straightforward to synthesize novel motion sequences from our grammar. We begin with an initial string consisting of a single non-terminal symbol. Then, we repeately rewrite each non-terminal symbol based on available rules until no non-terminal symbols are remained in the string. For example, let us consider that the string is $\mathbf{z} = \cdots X_i \cdots$ at a certain step, where $X_i \in V$. Any rules $r \in R$ can be a candidate rule for rewriting X_i if r has a form of $X_i \rightarrow \gamma$, where γ is a sequence of terminal and non-terminal symbols. Replacing X_i with the right-hand side γ of a selected rule expands the original string into a new one $\mathbf{z}' = \cdots \gamma \cdots$.

Note that if the rules in a given grammar G have at least one cyclic reference, the number of new strings that can be produced by G is unlimited. As a simple example, any self-referencing rules $r : X_i \rightarrow \cdots X_i \cdots$ allow us to recursively rewrite X_i an arbitrary number of times. In our experiments, we found more than one such cyclic references in learned grammars so that the grammars could produce arbitrarily many new strings.

Each terminal in the string corresponds to a set of segments of motions. Thus, we need to choose a single segment for each terminal and stitch every selected segment sequentially into a single, eventual motion. High-quality motions can be obtained at this last stage by carefully selecting a series of segments such that each segment has a smooth connectivity with its adjacent other segments. Given a pair of segments, we use the pose-to-pose distance function $\mathrm{d}(\mathbf{x}_t, \mathbf{x}'_t)$ to check if the end pose of one segment and the start pose of the other segment are similar enough to be concatenated smoothly. Because the number of combinations for segment selection grows exponentially with the string length, it is not practical to enumerate every possibility and pick the best one. Instead,

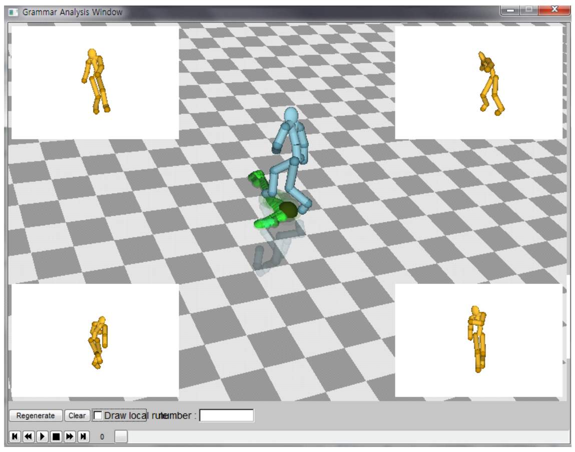

Figure 4: *Our interactive animation authoring interface. The four small windows at corners show possible new sequences that can replace the user-selected region, which is displayed as translucent spheres over the root trajectory.*

we use a greedy approach to find a reasonable approximation. An arbitrary segment is selected initially for the first terminal in the string. Then, we proceed in turn to select the locally optimal segments such that each segment has the best connectivity with its previous segment.

Any randomly generated motions in this way share the same syntactic structures at least locally with the input motion data. For more controllability over synthesized motions, we provide an animation authoring interface that allows the user to interactively compose his/her intended motion sequences under the syntactic constraints that are automatically imposed by learned grammars (see Figure 4). The user is initially provided with a randomly generated motion sequence. We display both the string representation and its synthesized motion simultaneously as a root trajectory of the motion that is visually divided by colors representing terminal symbols. When the user selects a terminal in the string, our system identifies which non-terminal symbol has generated the selected terminal, and presents the user with a set of alternative motion sequences that can replace its embracing region by randomly re-generating substrings from the identified non-terminal symbol. Choosing one of the alternatives removes the original substring, and inserts the newly generated substring into the region. Additionally, the editing region can be expanded or shrunken by finding the ancestor nodes of the selected terminal at a higher or lower level in the parse tree of the edited string.

7. Experimental Results

For our overall experiments, we used a C++ implementation of the ABL that is freely available with the grant of its authors. The library is efficient enough to scale well with the size of training data. As its key options, we used *suffix tree 3* (-a st3) for the alignment phase, and *branch* (-s b) in the selection phase. All of our timing data in this section was measured on a desktop PC with Intel 3.2 GHz CPU.

We first tested the basic capability of the ABL by manullay defining a simple grammar that describes the behavior of a basketball player, randomly generating a set of strings from the grammar, and learning a new grammar from the set of strings. The terminal set of the test grammar consists of r (receive), d (dribble), p (pass), s (shoot), f (fake), and m (move without the ball). Five non-terminals are associated with a set of production rules $S \to rX_1X_2m$, $X_1 \to X_3X_4X_3$, $X_2 \to p|s$, $X_3 \to f|\varnothing$, and $X_4 \to d|\varnothing$. We randomly generated 6 strings from the grammar: $rfdpm$, $rfdsm$, $rdfsm$, $rdpm$, $rdsm$, and rpm. From those strings, the ABL could induce a reasonably compact new grammar consisting of 5 non-terminal symbols and 11 rules.

We experimented with a variety of human motion data including break-dancing, boxing, and basketball (see Table 1). For all of the motions, we identified terminal symbols in a hierarchical manner that split each motion into several subsequences, run the genetic algorithm for initial segmentation of each subsequence, and finally performed the simulated annealing algorithm for finely tuning the segmentation of the entire sequence. In the phase of the genetic algorithm, we fixed the population size and the number of generations to 100 and 500, respectively. For break-dancing, the computation time for each generation was about 0.5 seconds. The simulated annealing step performed 1000 iterations for local adjustment of intervals, which took about four minutes. In order to examine the effect of the number of terminal symbols on the quality of learned grammars, we run our method over various values of K. Although we have not yet derived a generalized rule of determining K, visual analysis roughly indicates that increasing the value of K improves the clarity of symbols while decreasing the flexibility of learned grammars. We currently regards the value of K as a balancing factor that can be experimentally determined according to the compactness and flexibility of the grammar to be pursued.

In order to demonstrate the benefit of learning context-free grammars, we performed a visual comparison of our approach with a finite state machine in which each node corresponded to a terminal symbol in the same way as in our approach. In contrast with our ability to learn hierarchical structures, the finite state machine encoded only linear structures by creating transitions between any two symbols if those consequently occurred in the string of given motion data. Although both methods synthesize smoothly connected motions, our method could be able to reproduce the original behaviors much more naturally than the finite state machine. Please refer to our accompanying video for side-by-side comparison between our approach and the finite state machine in detail.

Input motion	# of frames	# of segments	# of terminals	# of rules
Break-dance	8677	367	20	128.6
			30	130
			40	133.2
			60	150.4
Boxing	12908	885	20	163.6
			30	137.6
			40	138.4
			60	157.2
Basketball	3198	156	12	8
			20	17

Table 1: *Experimental data. The number of rules is the mean value obtained from several different sets of sentences.*

Break-dancing captures a free-style performance of a professional b-boy, which includes various combinations of elementary break-dance moves such as top rocks, floor rocks, power moves, and freezes (see Figure 1). Its syntactic structure is not obvious enough to be easily recognized just by observing the motion carefully. The evidence it has robust composition rules is clearly revealed when the finite state machine imitates the performance; the dancer seems to have little knowledge about how to connect several basic moves plausibly. On the other side, we do not feel such a sense of disharmony from an arbitrarily synthesized motion produced by our approach.

Boxing also comprises a broad variety of combinations of footworks and punches performed by a professional boxer (see Figure 2). This motion data consists of the largest number of frames among our experimental data sets, but our two-level process of terminal identification could have found a reasonable collection of elementary actions. Similarly as in break-dancing, our method generated more natural-looking combinations of the basic moves than the finite state machine.

Basketball differs from other motions in that its sentential structure is recognized clearly as a sequence of preparation, dribbling, shooting, and just walking for a while (see Figure 3). Because we intended that our grammar could reproduce such a structure once in each new string, we manually divided a long sequence of repeated cycles into 13 shooting clips instead of using our automatic process. Although our grammar still produced better results than the finite state machine, it often generated more than a single shot against our expectation due to its cyclic references among non-terminals. We have not yet found a clear explanation about this result, but guess that the number of training examples is too small for the ABL to identify a common grammatical structure.

8. Discussion

We have presented a new motion synthesis method that rearranges captured motion data in the time domain while preserving the syntactic structures embedded in the original motion. To our knowledge, this method is the first to identify syntactic structures from human motion based on the context-free grammar, as well as to synthesize novel motions that preserve learned syntactic structures.

The primary benefit of our approach is that complex, and often hierarchical, human movement patterns can be automatically analyzed from raw motion capture data, and compactly encoded into a grammatical form. Once such grammatical structures have been obtained, we can easily create a large variation on the original motion data without destroying its implicit rules of ordering basic movements. In addition to the purpose of animation authoring, we expect that our method can be utilized to various other areas including performance choreography, sports analysis, and gestural communication.

Transforming the continuous domain of motion data into the discrete domain of symbolic representation plays the central role in our approach, but gives rise to some limitations at the same time. A key drawback is the possible mismatch between symbolic and perceptual equivalence of movements. In other words, a single symbol can represent perceptually different movements together, and two or more different symbols can represent perceptually similar movements. Investigating perceptually meaningful criteria for clustering movements would be an important research direction.

Our grammar-based approach currently does not allow the user to impose various spatial and temporal constraints such as intended trajectories, target locations of end effectors, and so on. Although this problem can be mitigated somewhat by interactive editing, it still is a significant limitation in comparison with the motion graphs that provide a compact search space for constrained motion synthesis. The key challenge in extending our approach to incorporate such functionality is that the solution space is composed of trees that can be produced by a context-free grammar, even allowing recursive structures. As one feasible approach to this problem, we plan to experiment with the genetic programming, which is a variation of the genetic algorithm specially tailored to cope with the solution space composed of trees.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0001148 and No. 2011-0018340). The second author was partly supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0004676).

References

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 402–408. 2

[BCvdP08] BEAUDOIN P., COROS S., VAN DE M., POULIN P. P.: Motion-motif graphs. In *SCA '08: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2008), pp. 117–126. 3

[BSP*04] BARBIČ J., SAFONOVA A., PAN J.-Y., FALOUTSOS C., HODGINS J. K., POLLARD N. S.: Segmenting motion capture data into distinct behaviors. In *GI '04: Proceedings of Graphics Interface 2004* (2004), pp. 185–194. 3

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29*, 4 (2010), 1–10. 3

[DGL09] DENG Z., GU Q., LI Q.: Perceptually consistent example-based human motion retrieval. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 191–198. 3

[dlH05] DE LA HIGUERA C.: A bibliographical study of grammatical inference. *Pattern Recognition 38* (2005), 1332–1348. 3

[FF05] FORBES K., FIUME E.: An efficient search algorithm for motion data using weighted pca. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 67–76. 3

[FMJ02] FOD A., MATARIC M. J., JENKINS O. C.: Automated derivation of primitives for movement classification. *Autonomous Robots 12*, 1 (2002), 39–54. 3

[Gus97] GUSFIELD D.: *Algorithms on strings, trees, and sequences*. Cambridge University Press, 1997. 7

[HG07] HECK R., GLEICHER M.: Parametric motion graphs. In *SI3D '07: Proceedings of the 2007 ACM Symposium on Interactive 3D Graphics* (2007), pp. 129–136. 2

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 214–224. 2

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

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *ACM Transactions on Graphics (SIGGRAPH 2002)* (2002), vol. 21, pp. 473–482. 3

[KS05] KWON T., SHIN S. Y.: Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 29–38. 3

[KTWZ10] KRÜGER B., TAUTGES J., WEBER A., ZINKE A.: Fast local and global similarity searches in large motion capture databases. In *SCA '10: Proceedings of the 2010 Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2010), pp. 1–10. 3

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3 (2002), 491–500. 2, 3, 4

[LK90] LEONARD KAUFMAN P. J. R.: *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990. 4

[LL06] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. *Graphical Models* (2006), 158–174. 2

[LPL11] LEE K. H., PARK J. P., LEE J.: Movement classes from human motion data. *LNCS Transactions on Edutainment* (2011). 5

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), pp. 39–48. 2

[LS00] LANGLEY P., STROMSTEN S.: Learning context-free grammars with a simplicity bias. In *Proceedings of the Eleventh European Conference on Machine Learning* (2000), Springer-Verlag, pp. 220–228. 6

[LZWM05] LIU G., ZHANG J., WANG W., MCMILLAN L.: A system for analyzing and indexing human-motion databases. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2005), SIGMOD '05, ACM, pp. 924–926. 3

[MR06] MÜLLER M., RÖDER T.: Motion templates for automatic classification and retrieval of motion capture data. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), pp. 137–146. 3

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

[MYHW08] MENG J., YUAN J., HANS M., WU Y.: Mining motifs from human motion. In *EUROGRAPHICS 2008–Short Papers* (2008), pp. 71–74. 3

[NMW97] NEVILL-MANNING C. G., WITTEN I. H.: Identifying hierarchical structure in sequences: a linear-time algorithm. *J. Artif. Int. Res. 7* (September 1997), 67–82. 6

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 106. 2

[SO06] SHIN H. J., OH H. S.: Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proceedings of the 2006 Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2006), pp. 291–298. 2

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. *ACM Transactions on Graphics 26*, 3 (2007), 7. 2

[vBM*10] ŠT'AVA O., BENEŠ B., MĚCH R., ALIAGA D. G., KRIŠTOF P.: Inverse procedural modeling by automatic generation of l-systems. *Computer Graphics Forum 29* (2010), 665–674. 3

[vZ00] VAN ZAANEN M.: Abl: alignment-based learning. In *Proceedings of the 18th conference on Computational linguistics - Volume 2* (2000), pp. 961–967. 3, 6